A TWO-PHASE DAMPED-EXPONENTIAL MODEL FOR SPEECH SYNTHESIS

THESIS
H. Allan Arb
Captain, USAF

AFIT/GE/ENG/96D-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GE/ENG/96D-02

A TWO-PHASE DAMPED-EXPONENTIAL MODEL FOR SPEECH SYNTHESIS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

H. Allan Arb, BSEE
Captain, USAF

December 17, 1996

Acknowledgements

I first need to thank God for giving me the skills, perseverance, and patience to complete this program. Secondly, I wish to extend a tremendous amount of thanks to my wife, Debbie, for supporting me and encouraging me throughout the last year and a half. Special gratitude goes out to my thesis committee: Dr. Marty DeSimio, Dr. Steve Rogers, and Dr. Tim Anderson. I owe a great debt of gratitude as well to my sponsor, Dr. Ray Slyh. Without his help, explanation of many of the models and concepts used, and critical evaluation of the writing and work done, this thesis would not have been possible. Finally, I need to thank the 4IGY's in class GE-96D. Without the help of my peers in the classes taken, I never could have made the achievements I did in this program. Thanks to you all.

H. Allan Arb

Table of Contents

List of Figures

List of Tables

Abstract

It is well known that there is room for improvement in the resultant quality of speech synthesizers in use today. This research focuses on the improvement of speech synthesis by analyzing various models for speech signals. An improvement in synthesis quality will benefit any system incorporating speech synthesis. Many synthesizers in use today use linear predictive coding (LPC) techniques and only use one set of vocal tract parameters per analysis frame or pitch period for pitch-synchronous synthesizers. This work is motivated by the two-phase analysis-synthesis model proposed by Krishnamurthy. In lieu of electroglottograph data for vocal tract model transition point determination, this work estimates this point directly from the speech signal. The work then evaluates the potential of the two-phase damped-exponential model for synthetic speech quality improvement. LPC and damped-exponential models are used for synthesis. Statistical analysis of data collected in a subjective listening test indicates a statistically significant improvement (at the 0.05 significance level) in quality using this two-phase damped-exponential model over single-phase LPC, single-phase damped-exponential, and two-phase LPC for the speakers, sentences, and model orders used. This subjective test shows the potential for quality improvement of synthesized speech and supports the need for further research and testing.

A TWO-PHASE DAMPED-EXPONENTIAL MODEL FOR SPEECH SYNTHESIS

I. Introduction

This thesis considers the problem of improving the quality of speech generated by speech synthesizers. Speech synthesis has several military and commercial applications including:

- Speech output for information systems.
- Speech warnings for aircraft and other machinery.
- Speech output for automatic reading systems for the blind.

Today, state-of-the-art synthesizers generate highly intelligible speech; however, modern speech synthesizers still cannot generate natural-sounding speech [23]. This thesis outlines several models for speech signals and documents some experiments that examine the effects of these models on synthetic speech quality.

1.1 Overview

Much of the past research on speech synthesis for military applications has focused on improving the intelligibility, not the quality, of the synthesized speech [20–22]. However, improving the quality of synthesized speech often also improves the intelligibility. This quality improvement was not of primary concern to much of the past research on speech synthesis for military applications. This thesis investigates the effects various models for speech signals have on the quality of synthesized speech.

Much of the current research focuses on the analysis of digitized speech waveforms, extraction of key parameters from the speech, and regeneration of the speech

signal based only on the extracted parameters and the chosen model for speech synthesis [2, 14–16, 20, 22, 23]. This process is called analysis-synthesis. Systems using analysis-synthesis reconstruct spoken words in the same order as stored speech. Another method for synthesizing speech is text-to-speech synthesis (TTS). TTS systems convert textual information into a speech signal based on stored characteristics of one or more speakers' voices. There are several stages within the TTS system of which the synthesis stage is one. Analysis-synthesis can be used to evaluate the effects of various signal models on the synthetic speech quality without the need for the additional stages and without their distorting effects. Despite the differences between analysis-synthesis and text-to-speech synthesis, improvements made using analysis-synthesis can be incorporated into the more complex TTS systems.

There are several models for analysis-synthesis of speech waveforms that produce telephone quality or better speech signals [3, 16, 23, 30]. There are still no speech synthesis systems capable of generating speech that is indistinguishable by humans from actual speech at reasonable expense. The problem involves "inadequate modeling of human speech production in coarticulation, intonation, and vocal-tract excitation" [23]. If we model speech production in these areas more accurately, we should be able to generate synthetic speech that is of higher quality than previous methods.

In [15], Krishnamurthy presented a technique for analysis-synthesis using a two-phase approach. His technique estimates the model parameters in two stages, or phases. First, it estimates the model parameters over the open glottal phase (when the vocal cords are open) and then again during the closed phase (when the vocal cords are closed). The synthetic speech waveform is generated after estimating parameters for both the excitation waveform and the vocal tract filter [15]. The difference between his proposal and systems already developed is the use of unique model parameters for both the open phase and the closed phase for the glottal

excitation as well as the vocal tract filter. Krishnamurthy's model is the motivating factor behind this research.

1.2 Problem

Most current speech synthesizers assume that speech characteristics are constant over an entire pitch period. This work investigates the quality of speech synthesized using a two-phase "Sum-of-Exponentials" model for synthesizing speech similar to the model proposed by Krishnamurthy [15]. This model allows for the vocal tract model parameters to vary over the two phases. It is hypothesized that this approach will allow for more natural sounding speech synthesis. This work is different than Krishnamurthy's in that the phase transition point and glottal closing instants will be estimated directly from the speech signal without the use of an electroglottograph sampled simultaneously with the original speech samples. In addition, complete sentences are analyzed and synthesized as opposed to the steady state vowels used by Krishnamurthy [15].

1.3 Definition of Terms

1. Pitch: The fundamental frequency (f_0) of a sound or speech signal [12] corresponding to the repetition rate of puffs of air exiting the vocal cords as they open and close.
2. Pitch Period: The fractional inverse of the pitch ($1/f_0$).
3. Glottis: The area between the vocal cords.
4. Closed Glottal Phase: The interval within a pitch period in which the vocal cords are together (i.e. the glottis is closed).
5. Open Glottal Phase: The interval within a pitch period in which the vocal cords are separated (i.e. the glottis is open).

6. LPC Synthesizer: A speech synthesis system that generates voiced speech by filtering a quasi-periodic excitation waveform with a linear, time-varying, autoregressive (all-pole) filter. The filter poles are determined using linear predictive coding (LPC) techniques. The system uses a broadband noise source as the filter input for unvoiced speech.

1.4 Assumptions

In this research, it is assumed that speech files are phonemically labeled as a function of sample index as identified by a group of experts. The TIMIT speech data base contains speech files labeled in this manner and will therefore be used [1].

1.5 Scope

Software development for speech analysis-synthesis consists of one- and two-phase damped-exponential models and one- and two-phase LPC models. Results and conclusions are drawn from data collected in subjective listening tests using 19 subjects. These tests compare the speech synthesis techniques using a one-phase damped-exponential model, a one-phase LPC model, the two-phase damped-exponential model, and a two-phase LPC model.

1.6 Approach/Methodology

First an analysis-synthesis system is developed using the sum-of-exponential (also termed "damped-exponential") model while applying the model to the entire pitch period. There will be no open and closed phase modeling in this first system. Next another analysis-system is developed using the two-phase model similar to that proposed by Krishnamurthy. Finally, the one- and two-phase systems are adapted for use with LPC techniques. Subjective listening tests are conducted and statistical analyses performed to evaluate the effects on quality of the speech signal models used in the analysis-synthesis systems.

1.7 Outline

An outline of this thesis is as follows. Chapter II provides brief synopsis of speech synthesis and key techniques. The chapter includes: a section on speech synthesizers from early vocoders to state-of-the-art systems; a review of various models proposed for the glottal source waveform; and the model developed by Krishnamurthy. Chapter III contains the description of the speech synthesis systems developed. Chapter IV describes the subjective listening test and statistical analyses performed, and presents the results obtained from these experiments. Finally, Chapter V provides a summary of the results and answers to the two fundamental questions for this thesis:

1. Does the use of a two-phase damped-exponential model in voiced-speech synthesis show improvement in quality over that obtained with one- and two-phase LPC or one-phase damped exponential models?
2. Does the use of a two-phase model in general show an improvement in quality over one-phase models?

   Answering these questions is the overall goal of this work.

II. Speech Synthesis Background

It is well known that the quality of synthetic speech is directly affected by the models chosen to generate it. Variations in the shape of the pulse used to represent the glottal excitation (volume velocity of air as it enters the glottis) from the actual shape cause degradations in the quality of the resulting speech waveform [3,8,9,23]. Another source of degradation in synthetic speech is the failure to model the coupling between the subglottal (lungs and trachea up to the glottis) and supraglottal (glottis to lips and nostrils) systems. Many models for speech assume that the characteristics of the vocal tract are fixed over a pitch period. However, the coupling between the subglottal and supraglottal systems causes a shift in formant frequencies from closing to opening phases and vice versa [15]. It is expected that speech synthesis using a model accounting for this coupling will enhance the natural quality of speech synthesis [15,23].

This chapter first gives a brief history of the development of speech synthesizers. It then introduces a model proposed by Krishnamurthy [15] which does account for the coupling between the subglottal and supraglottal systems.

2.1 Historical Perspective

The scientific study of speech began during the Renaissance when mechanical models were first constructed to imitate speech. Mechanical speech synthesis was first well documented in St. Petersburg and Vienna in the late 18^{th} century [6]. However, the beginning of the era of modern speech technology is considered to be the 1930's. In 1939, Dudley introduced the vocoder (voice coder) which led to the idea of parametric speech representation and coding. And so began an explosion in speech research [12].

In 1964, Holmes presented a speech synthesis system that generated an analog speech signal by adding a small number of sinusoids together [11]. This system was

one of the first of a class of synthesizers called formant synthesizers. The system determines the number, frequency, amplitude, and duration of each sinusoid using a set of rules for each sound to be synthesized.

Between 1964 and 1973 a class of algorithms called LPC (linear predictive coding) vocoders were developed. Basic LPC vocoders generate a speech signal by modeling the glottal excitation as either a quasi-periodic train of impulses for voiced speech or random noise for unvoiced speech. LPC vocoders filter the excitation with an all-pole linear filter with poles corresponding to the formant resonances of the vocal tract. The resultant synthetic waveform is highly intelligible, but very unnatural in sound.

In an effort to improve the quality of speech synthesized by LPC techniques, Rosenberg investigated various models for the glottal excitation [27]. He found that both trigonometric and polynomial pulse shapes improve speech quality relative to that obtained with simple impulse excitations. Using Rosenberg's glottal pulse shapes, Holmes was able to improve the quality of speech produced by his formant synthesizer [10].

Several scientists have developed similar models for glottal excitation [8, 9, 14] while others developed models for the derivative of the glottal pulse [5, 7]. These models laid the foundation for the development of state-of-the-art synthesizers such as Digital Equipment Corporation's DECtalk. AT&T has also produced a highly-intelligible text-to-speech synthesizer based on glottal pulse excitations for voiced speech [30]. Still, these state-of-the-art synthesizers do not produce natural-sounding speech.

2.2 Krishnamurthy's Two-Phase Model

In 1992, Krishnamurthy developed an algorithm for jointly determining the resonances of the vocal tract and the parameters of the glottal source waveform [15]. The algorithm uses the Liljiencrants-Fant (LF) glottal-flow derivative model [5] as

the effective source to the vocal tract filter. The algorithm also allows for the coupling between the subglottal and supraglottal systems during the open phase of a pitch period by allowing different filter models for each phase. Krishnamurthy's algorithm provides the motivation for this research. This section of the review describes the models and analysis procedure in more detail.

2.2.1 Speech Production Model. Figure 1 illustrates the model of speech production used in this analysis. The vocal tract filter $V(z)$ is the transfer function relating the volume velocity at the lips to the glottal volume velocity. The input to this filter is the effective voice source, $q(n)$ defined as the differentiated glottal volume velocity. The effective voice source models the effects of the lip radiation as well as the glottal volume velocity. The output of the filter, $s(n)$ is the radiated speech pressure wave (sampled at a microphone at time index n) [15].

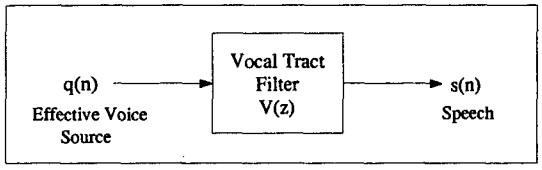

Figure 1 Speech production model.

2.2.1.1 Effective Voice Source Model. The effective voice source over one pitch period is modeled with a discrete-time version of the model proposed by Liljiencrants and Fant (LF) [5]. Realizing that the radiation through the lips can be modeled as a first order filter, they derived a model for the derivative of the glottal volume velocity [5]. The LF model couples the volume velocity pulse with the radiation effects of the lips. The continuous-time mathematical form of the model is

$$u_g'(t) = \begin{cases} E_0 e^{\alpha t} \sin(\omega_g(t)), & 0 \le t \le T_e \\ -\left(\dfrac{E_e}{\epsilon T_e}\right)\left(e^{-\epsilon(t-T_e)} - e^{-\epsilon(T_c-T_e)}\right), & T_e \le t \le T_c \end{cases} \quad (1)$$

A sample pulse derivative waveform is illustrated in Figure 2.

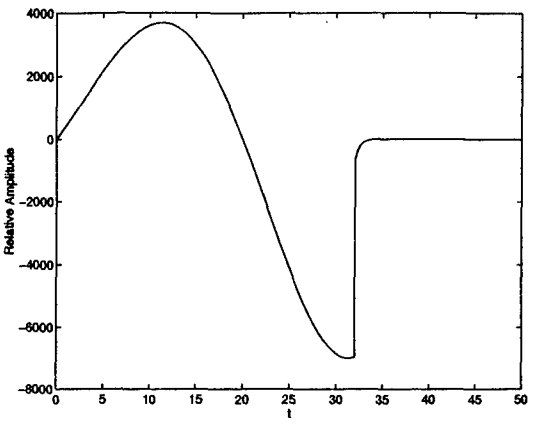

Figure 2 Liljiencrants-Fant glottal flow derivative waveform for parameter set E_0=2622.799, α=0.032, f_g=0.025, T_e=32, ϵ=2.

Krishnamurthy's discrete version of the LF model is defined as

$$q(n) = \begin{cases} A_{go} e^{\alpha_{go} n} \sin(\omega_{go} n + \phi_{go}), & n = 0, \ldots, N-1 \\ -A_{gc} e^{-\alpha_{gc}(n-N)}, & n = N, \ldots, M-1 \end{cases} \tag{2}$$

where the subscripts *go* and *gc* identify the distinction in parameters between the open and closed phases respectively. The intervals $[0, N-1]$ and $[N, M-1]$ represent the open and closed phases. The phase angle, ϕ_{go}, accounts for discrepancies between the actual glottal opening instant and the estimated opening instant. Rewritten using complex exponentials, (2) becomes

$$q(n) = \begin{cases} C_o z_{go}^n + C_o^* (z_{go}^*)^n, & n = 0, \ldots, N-1 \\ C_c z_{gc}^{n-N}, & n = N, \ldots, M-1 \end{cases} \tag{3}$$

where * denotes a complex conjugation and

$$C_o = 0.5 A_{go} e^{j(\phi_{go} - \pi/2)}, \qquad z_{go} = e^{\alpha_{go} + j\omega_{go}}, \qquad C_c = -A_{gc}, \qquad z_{gc} = e^{-\alpha_{gc}} \tag{4}$$

2.2.1.2 Vocal Tract Model. Krishnamurthy models the vocal tract filter as a pole-zero system as opposed to the all-pole model used in LPC synthesis.

Furthermore, the model allows the transfer function to change from closed phase to open phase within a pitch period. The filter transfer function over one entire pitch period is defined as

$$V(z) = \begin{cases} V_c(z) = \dfrac{B_c(z)}{A_c(z)}, & \text{during closed phase} \\ V_o(z) = \dfrac{B_o(z)}{A_o(z)}, & \text{during open phase} \end{cases} \tag{5}$$

where the denominator polynomials in each phase are defined as

$$A_c(z) = \prod_{i=1}^{K_c} [1 - z_c(i)z^{-1}][1 - z_c^*(i)z^{-1}] \tag{6}$$

$$A_o(z) = \prod_{i=1}^{K_o} [1 - z_o(i)z^{-1}][1 - z_o^*(i)z^{-1}] \tag{7}$$

The parameters K_c and K_o represent the number of complex conjugate pairs of poles estimated in the closed and open phases respectively. These pairs of poles represent the formant resonances of the vocal tract and can be used in a formant synthesizer.

The numerators ($B_c(z)$, $B_o(z)$), or "zeros", of (5) contribute to the complex amplitudes of the poles.

2.2.1.3 Speech Signal Model. Assuming that the effective voice source and vocal tract filters are modeled as described in the previous two sections, Krishnamurthy shows that the resulting speech signal, the output of the vocal tract filter, is the sum of exponential signals:

$$\hat{s}(n) = \begin{cases} B_{go} z_{go}^n + \cdots \\ B_{go}^*(z_{go}^*) + \sum_{i=1}^{K_o} [B_o(i)\{z_o(i)\}^n + B_o^*(i)\{z_o^*(i)\}^n], & n = 0, \ldots, N-1 \\ \\ B_{gc} z_{gc}^{(n-N)} + \cdots \\ \sum_{i=1}^{K_c} [B_c(i)\{z_c(i)\}^{n-N} + B_c^*(i)\{z_c^*(i)\}^{n-N}], & n = N, \ldots, M-1 \end{cases} \tag{8}$$

The $B_o(i)$ and $B_c(i)$ terms are complex amplitudes and B_{go} and B_{gc} are real amplitudes.

2.2.2 Analysis Procedure. Krishnamurthy begins his analysis by identifying the opening and closing instants within each pitch period with an electroglottograph (EGG) signal sampled simultaneously with the speech signal. Once these instants are determined, the parameters of the vocal tract filter are estimated over each phase in a two step process. First, the pole locations are estimated using a backward prediction procedure introduced by Parthasarathy, Kumaresan, and Tufts [18, 24]. Second, the complex amplitude parameters are estimated by solving a set of linear equations using the complex poles estimated in the first step.

Once these parameters are estimated, the speech signal can be resynthesized using (8). Krishnamurthy presents results comparing short segments of the original voiced speech signal to a synthetic signal generated using this procedure. He has shown that the sample-to-sample difference between the two waveforms is extremely low and definitely improved over the one phase modeling of conventional speech synthesizers. However, no listening tests were performed to assess the perceived quality by a human subject.

2.3 Summary

This chapter presented a brief history of speech synthesis from early formant and LPC synthesizers to glottal pulse-based systems. The literature is consistent in identifying the need for higher quality, more natural synthetic speech. State-of-the-art synthesizers, while producing highly intelligible synthetic speech, still cannot produce natural sounding speech. While models for glottal excitation commonly allow for more than one phase, few vocal tract models reported in the literature allow for multiple phases within a pitch-period. The model proposed in [15] by Krishnamurthy is one model for speech that does allow for both a multi-phase excitation

and a multi-phase vocal tract. Krishnamurthy showed that the difference between synthetic and original signals can be minimized for vowel sounds using his two phase model. This evidence supports the idea that speech synthesis in general can be improved with his procedure and that further research into applying it to larger speech segments is warranted.

III. Analysis-Synthesis Systems

This chapter presents the methods used to analyze and re-synthesize the TIMIT speech samples. Figures 3 and 4 outline the processes used for analysis-synthesis of the damped-exponential and LPC methods respectively. These figures also serve as an outline for this chapter. First, the speech sample must be decomposed into analysis frames. Pitchmarks (instants of glottal closure) are estimated for voiced speech, and segments of unvoiced speech are decomposed into small constant length analysis frames. Second, the phoneme label corresponding to each analysis frame is compared to values in a table to determine whether or not the frame is voiced. Based on the results of the voicing analysis and the synthesis model (one-phase LPC, two-phase damped-exponential, etc.), the system estimates model parameters and synthesizes a new waveform. The MATLAB® code to implement these methods is provided in Appendix B.

3.1 Pitchmark Estimation

The first step in the analysis-synthesis process is to decompose the input speech waveform into analysis frames roughly equal to one pitch period. To accomplish this, these analysis-synthesis systems use two programs contained in the "Entropic Signal Processing System" (ESPS) from Entropic Research Laboratory, Inc. Specifically, they use the *get_f0* and *epochs* programs to generate a data file containing non-zero values only at estimated glottal closing instants.

Get_f0 uses an algorithm similar to the method of Secrest and Doddington [28]. This algorithm estimates the fundamental frequency (f_0) using the cross correlation function and dynamic programming. It operates on a standard sampled-data file formated for use with ESPS and produces a file containing estimates of: f_0; a voicing probability; the RMS energy value; and peak autocorrelation value. *Epochs* uses these parameters, the sampled data file, and dynamic programming to determine

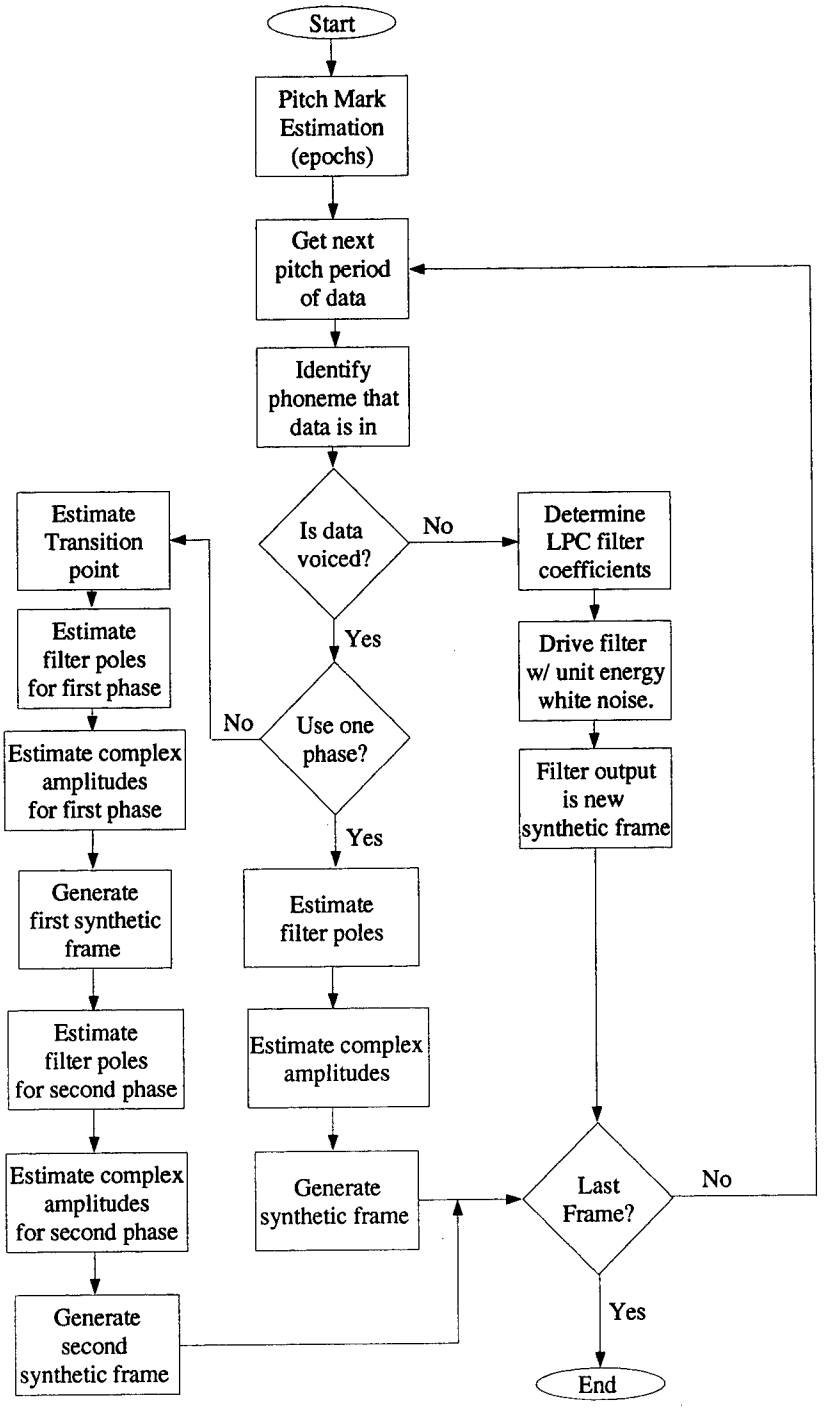

Figure 3 Analysis-Synthesis procedure for the damped-exponential model.

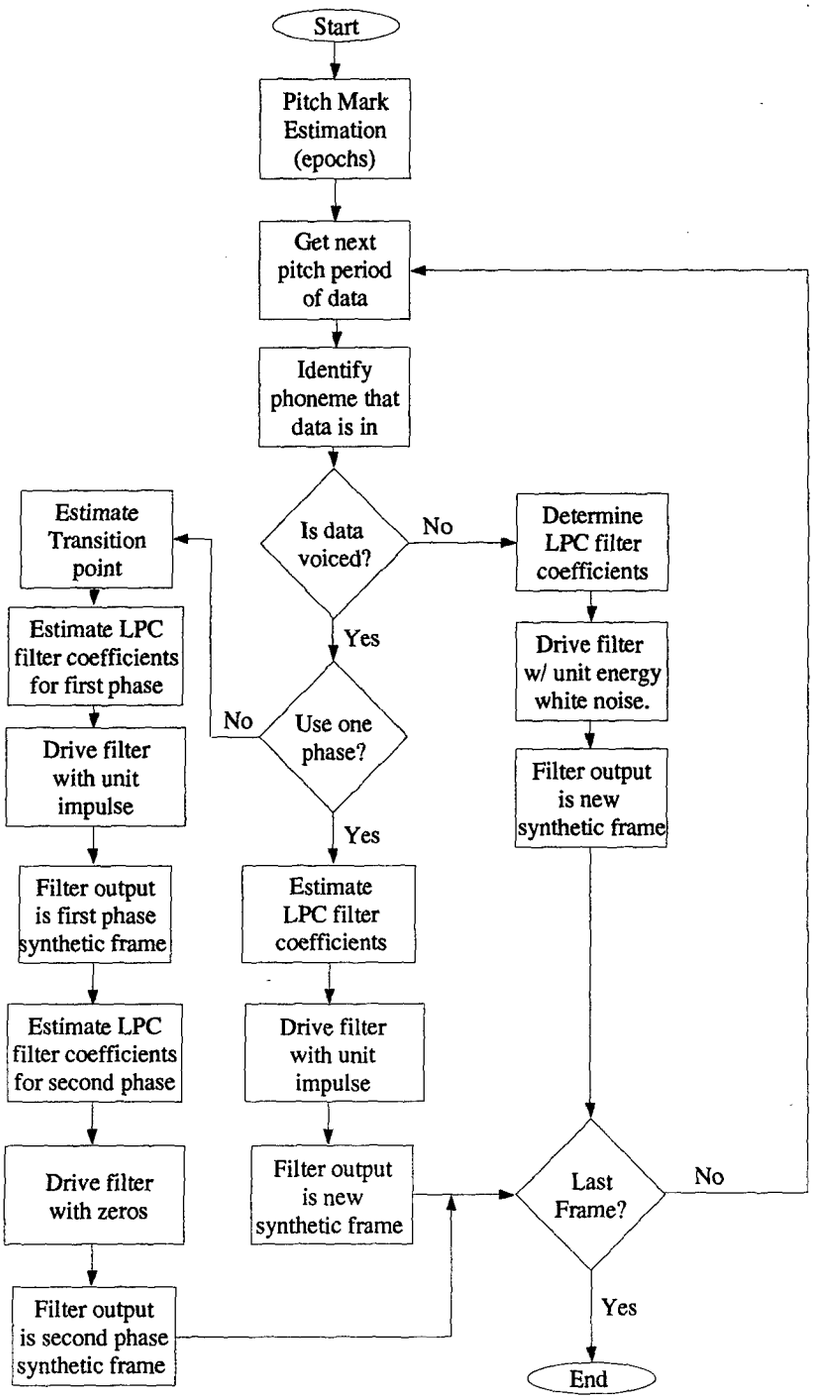

Figure 4 Analysis-Synthesis procedure for the LPC model.

the estimates of vocal fold closure instants. It produces a file with impulses at the estimated vocal fold closure instants and zeros elsewhere [4].

After *epochs* generates the pitchmark file, the analysis-synthesis system generates a two-column matrix containing the starting point and ending point of each analysis frame. In areas where the spacing between pitchmarks is larger than the expected maximum pitch period (e.g. unvoiced regions), the system divides the area into equal length analysis frames. The user selects the frequency, in hertz, defining the length of these divisions. For this work, the system used a 100 Hz frame rate (160 samples for a 16 kHz sample rate). Each pitchmark is moved to one sample prior to the nearest zero crossing in the speech signal. Since the pitchmarks estimated by *epochs* typically occur at large peaks in the speech file, estimating model parameters for a frame from one large peak to another produces an unstable model. This movement stabilizes the estimated model parameters. In addition, the movement of the pitchmarks reduces the error at frame boundaries.

3.2 Voicing Determination

Table 1 contains a list of each phoneme in the TIMIT corpus, its class (vowel, nasal, fricative, stop, etc.), and a voiced or unvoiced classification. This table is used to decide if an analysis frame is voiced or unvoiced.

The system must first determine to what phoneme the current analysis frame corresponds. The system compares the starting index of the frame within the original sentence to the starting and ending indices in the TIMIT phonemically labeled file for the sentence under analysis. It then identifies the phoneme label corresponding to the indices which contain the frame start index as the phoneme under investigation.

The system then searches the phoneme table until the phoneme label under analysis is found. The corresponding voiced/unvoiced classification is used as the voicing determination.

Table 1 Phoneme Voicing Determination

Phoneme	Class	Voicing	Phoneme	Class	Voicing
iy	Vowel	VOICED	ih	Vowel	VOICED
eh	Vowel	VOICED	ae	Vowel	VOICED
ix	Vowel	VOICED	ax	Vowel	VOICED
ah	Vowel	VOICED	ax-h	Vowel	VOICED
uw	Vowel	VOICED	ux	Vowel	VOICED
uh	Vowel	VOICED	ao	Vowel	VOICED
aa	Vowel	VOICED	ey	Vowel	VOICED
ay	Vowel	VOICED	oy	Vowel	VOICED
aw	Vowel	VOICED	ow	Vowel	VOICED
er	Vowel	VOICED	axr	Vowel	VOICED
m	Nasal	VOICED	em	Nasal	VOICED
n	Nasal	VOICED	nx	Nasal	VOICED
en	Nasal	VOICED	ng	Nasal	VOICED
eng	Nasal	VOICED	l	Liquid	VOICED
el	Liquid	VOICED	r	Liquid	VOICED
y	Liquid	VOICED	w	Liquid	VOICED
hh	Liquid	UNVOICED	hv	Liquid	VOICED
ch	Fricative	UNVOICED	jh	Fricative	UNVOICED
dh	Fricative	VOICED	z	Fricative	UNVOICED
zh	Fricative	UNVOICED	v	Fricative	VOICED
f	Fricative	UNVOICED	th	Fricative	UNVOICED
s	Fricative	UNVOICED	sh	Fricative	UNVOICED
b	Stop	VOICED	d	Stop	VOICED
dx	Stop	UNVOICED	g	Stop	VOICED
p	Stop	UNVOICED	t	Stop	UNVOICED
k	Stop	UNVOICED	pcl	Silence	UNVOICED
tcl	Silence	UNVOICED	kcl	Silence	UNVOICED
bcl	Silence	UNVOICED	dcl	Silence	UNVOICED
gcl	Silence	UNVOICED	epi	Silence	UNVOICED
h#	Silence	UNVOICED	pau	Silence	UNVOICED
q	Undefined	UNVOICED	und	Undefined	UNVOICED

It must be noted that the analysis-synthesis systems developed here do not rely on the knowledge of the phoneme containing each frame; only the voicing of each frame is important. The systems use the phoneme identification since the phonemically labeled files were conveniently provided with the TIMIT data base.

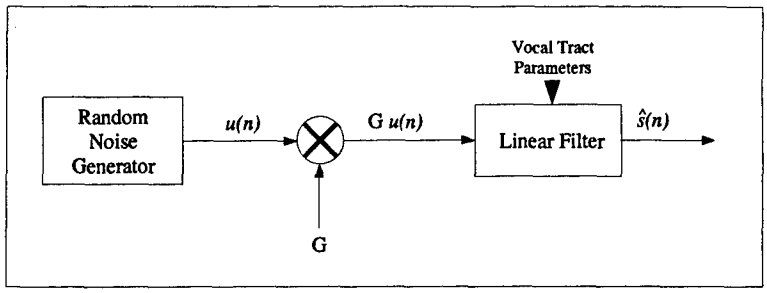

Figure 5 Linear prediction model for a single frame of unvoiced speech

3.3 Unvoiced Frame Synthesis

Noise driven LPC is used to re-synthesize unvoiced frames for all synthesis systems. Figure 5 illustrates the model used to synthesize an unvoiced frame. A unit energy random noise generator provides the input to an all-pole, linear filter. This input is weighted by G to provide the excitation waveform $Gu(n)$. The filter output at index n can be represented as a weighted sum of the past p outputs:

$$\hat{s}(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n) \tag{9}$$

where a_i is the i^{th} linear predictor coefficient and p is the system model order specified by the user. The synthesis system uses this equation to generate the new unvoiced frame after all parameters have been estimated.

3.3.1 LPC filter coefficients. It can be shown [26] that the LPC analysis equations for the covariance method are

$$\begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,p) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,p) \\ \phi(3,1) & \phi(3,2) & \cdots & \phi(3,p) \\ \vdots & \vdots & & \vdots \\ \phi(p,1) & \phi(p,2) & \cdots & \phi(p,p) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} \phi(1,0) \\ \phi(2,0) \\ \phi(3,0) \\ \vdots \\ \phi(p,0) \end{bmatrix} \tag{10}$$

where

$$\phi(i,k) = \sum_{m=0}^{N-1} s(m-i)s(m-k), \qquad \begin{matrix} i = 1 \ldots p \\ k = 0 \ldots p \end{matrix} \tag{11}$$

Equation (10) can be rewritten as

$$\mathbf{\Phi \hat{a}} = \boldsymbol{\phi} \tag{12}$$

The analysis system estimates $\mathbf{\hat{a}}$ in (12) by

$$\mathbf{\hat{a}} = \mathbf{\Phi}^{\dagger}\boldsymbol{\phi} \tag{13}$$

where † represents the pseudoinverse operation. The pseudoinverse is used instead of the inverse to handle any ill-conditioning of $\mathbf{\Phi}$.

3.3.2 Determination of gain (G). The gain (G) used to weight the random noise is defined as the square root of the minimum mean-squared error between the estimated filter output and the original waveform ($\hat{E}$):

$$G = \sqrt{\hat{E}} \tag{14}$$

where $\hat{E}$ is expressed as [26]

$$\hat{E} = \sum_{m=1}^{N} s^2(m) - \sum_{k=1}^{p} \hat{a}_k \sum_{m=1}^{N} s(m)s(m-k) \tag{15}$$

$$= \phi(0,0) - \sum_{k=1}^{p} \hat{a}_k \phi(0,k) \tag{16}$$

3.4 Damped-Exponential Voiced Frame Synthesis

The damped-exponential model represents the voiced waveform as

$$\hat{s}(n) = \sum_{i=1}^{p} A_i \rho_i^n \tag{17}$$

where A_i and ρ_i are the i^th complex amplitude and complex pole respectively. Each pole (if not real) occurs with its complex conjugate as well. This conjugate pairing results in the sinusoidal component resonating at the pole frequency. Appendix A gives a detailed description of these parameters and the method used to estimate them.

The single-phase damped-exponential system uses the same parameters for the entire pitch period. The two-phase damped-exponential system allows the parameters to change within the pitch period; thus a transition point must be estimated. After this point is determined, the pitch period is divided into two subframes corresponding to the first and second phases. Then, each subframe is synthesized using the same method as for a single model per pitch period. The second subframe is appended to the first to create the total synthetic pitch period.

3.4.1 Transition Point Determination. Most speech synthesis systems in use today only allow for one set of parameters to model the vocal tract resonances in each pitch period. The systems developed in this work, however, allow the parameters to change once within the pitch period in hopes of better modeling the effect of subglottal to supraglottal coupling when the vocal folds open. Typically, those systems that do allow for the parameters to change require identification of the opening instant using an electroglottograph (EGG) sampled simultaneously with the digitized speech. The analysis-synthesis systems presented in this thesis estimate the point at which the model parameters should change directly from the digitized waveform.

Model parameters are estimated over a small segment at the beginning of the pitch period (when the vocal cords are most likely closed) and again over a small segment at the end (when the vocal cords are most likely still open). The systems synthesize a complete pitch period using each set of model parameters and evaluate the error between each synthesized frame and the original frame to find the transition point. The error should be small near the area over which the parameters were estimated and larger elsewhere. The plots of the error between each synthesized frame and the original frame should cross at some point. This point is identified as the parameter transition point within the original pitch period.

3.4.1.1 Subframe parameter estimation and synthesis. The first subframe (small segment at the beginning of the pitch period) is defined as the initial $\frac{1}{3}$ of the pitch period. This fraction was chosen in an attempt to avoid including the actual transition point in one of the analysis windows. The second subframe (small segment at the end of the pitch period) is defined as the last $\frac{1}{3}$ of the pitch period. If the length of either subframe is less than or equal to the model order for its corresponding phase, the length is extended to be either 1.5 times the model order or 80% of the pitch period, whichever is smaller. This amount of extension was chosen to allow for an adequate amount of data from which to estimate the model parameters. Figure 6 illustrates a sample pitch period and its two subframes.

The system estimates model parameters (complex pole and amplitudes) over the first subframe in the same manner as described in Appendix A. However, instead of generating a synthetic frame the same length as the analysis window, it synthesizes a complete pitch period. Figure 7 illustrates the number of samples analyzed and the number of samples synthesized for each subframe analysis-synthesis.

Next, the system must estimate a new set of model parameters over the second subframe. Once these are complete, it also synthesizes a new complete pitch period. As in the normal synthesis method, the initial sample index of the analysis window

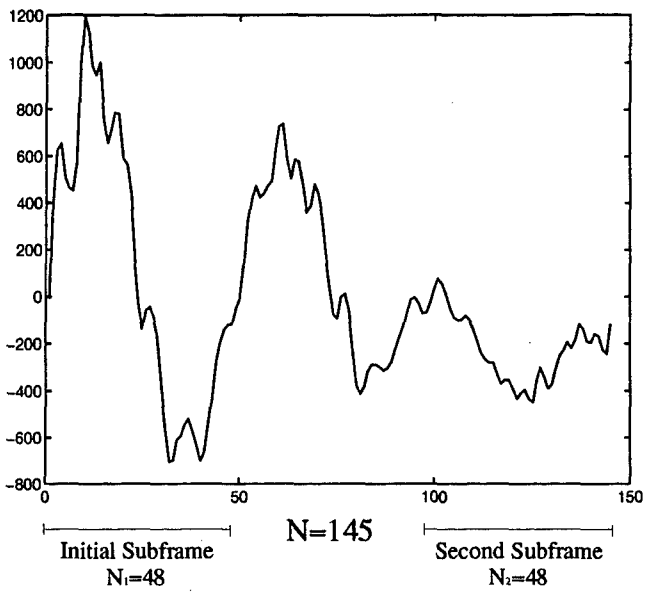

Figure 6 Plot of a sample pitch period and the subframes of $\frac{1}{3}$ the length.

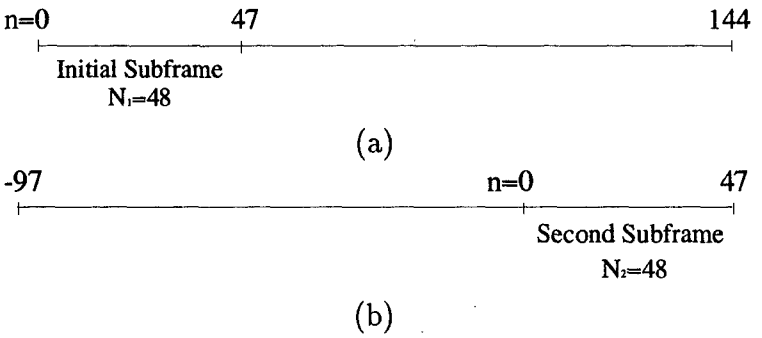

Figure 7 Illustrations of sample indexing for synthesis using model parameters estimated over the first subframe (a) and second subframe (b). The total number of samples synthesized is $N = 145$.

is considered to be $n = 0$. However, since the analysis window falls at the end of the pitch period, the system must synthesize the previous $\frac{2}{3}$ of the pitch period as negative sample indices as illustrated by Figure 7(b).

3.4.1.2 Error analysis and transition determination. Now that the two synthetic pitch periods have been generated, the system needs to analyze the error between each and the original waveform. The error is simply the normalized squared point-by-point difference between the original waveform and the synthetic frame:

$$e(n) = \frac{(s(n) - \hat{s}(n))^2}{\sum_{i=1}^{N}(s(i) - \hat{s}(i))^2} \tag{18}$$

Peaks of the error waveform ($e(n)$) are found using the peak selection algorithm given in McMillan [22]. The error amplitudes between peaks are linearly interpolated to generate a smoothed error waveform. Figure 8(a) shows the results of this procedure for the pitch period generated using model parameters from the first subframe of the sample data in Figure 7. Figure 8(b) shows the results for the second subframe synthesis.

Once the system generates the two smoothed error waveforms, it identifies the first crossing occurring between 20% and 60% of the pitch period length. In [24], Parthasarathy and Tufts give a typical range for opening instants as 20-50% of the pitch period. Empirical testing showed the extension of this range to 60% resulted in better quality as many transition points were located between 50% and 60%. Figure 8(c) shows the two smoothed error waveforms overlayed. The estimated transition point is clearly labeled.

3.5 LPC Voiced Frame Synthesis

Figure 9 illustrates the linear systems used to model voiced pitch periods with LPC. Figure 9(a) shows that a weighted impulse at $n = 1$ is used as the input to the linear vocal tract filter when a single phase per pitch period is desired. Figure 9(b)

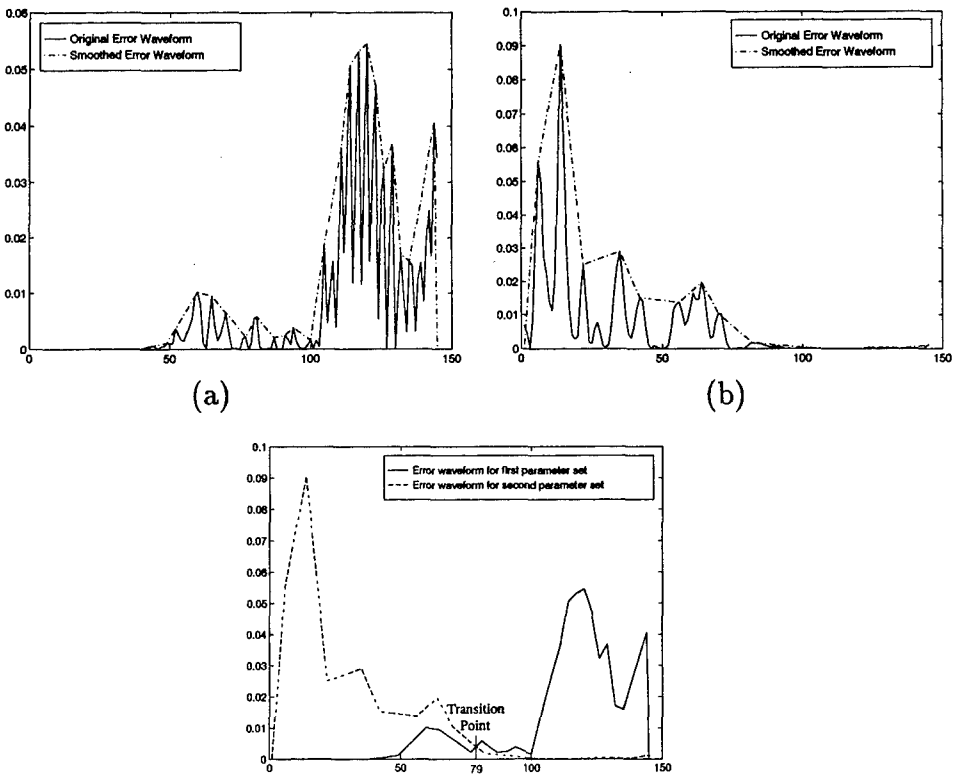

Figure 8 Plots of the error waveform (original and smoothed) for the synthetic waveform generated with parameters estimated over (a) the first subframe and (b) the second subframe. (c) is a plot of the two smoothed waveforms overlayed with the estimated transition point labeled.

shows that the same input is used for a two-phase approach, but the filter parameters are allowed to change. Note that the input to the filter is a sequence of zeros for the second phase. Also, for voiced LPC speech synthesis, the last p (p = model order) synthesized samples of the previous analysis frame are used as initial conditions for the vocal tract filter in the current analysis frame.

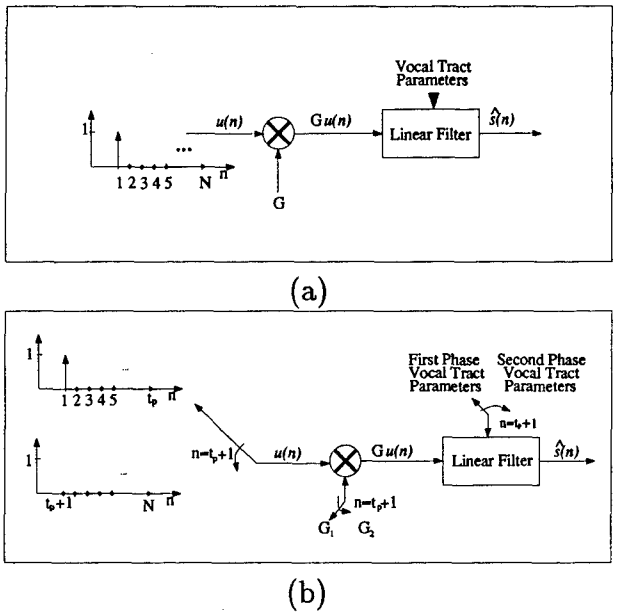

Figure 9 Linear prediction model for a single pitch period of voiced speech using (a) one phase per pitch period and (b) two phases.

The LPC analysis-synthesis systems estimate the parameters for the vocal tract filter and gain in the same manner as described in section 3.3. For a two-phase approach, the transition point is estimated as described in section 3.4.1 and the gain and LPC coefficients are re-estimated in the second frame. Thus, the two-phase LPC system and the two-phase damped-exponential system use the same transition points for each sentence and speaker; only the voiced speech synthesis model differs between the systems.

3.6 Summary

This chapter described the analysis-synthesis systems used in this thesis. An overview of the entire analysis-synthesis process was illustrated in Figures 3 and 4. While many analysis-synthesis systems in use today make use of LPC and some the damped-exponential model, most use only one set of vocal tract parameters per pitch period. Some systems do use two phases per pitch period, but require synchronous EEG data to estimate the vocal tract parameter transition point. This chapter described a method to estimate this point directly from the digitized speech sample. There are no known analysis-synthesis systems in use today that employ this technique. The next chapter discusses the experiments that were conducted to test the relative merits of the two-phase damped-exponential model as compared to the one-phase LPC, one-phase damped-exponential, and two-phase LPC models.

IV. Experiments and Results

This chapter describes the design and implementation of the subjective listening test used to evaluate the algorithms discussed in Chapter III. In particular, this chapter details the test sentences used, the model orders chosen, and the statistical analyses that were performed on the data collected.

4.1 Subjective Listening Test

A panel of 19 volunteers from the Air Force Institute of Technology and the USAF's Armstrong Laboratory (AL) was formed under AL's Human Use Review Committee (HURC) protocol #83-58: "Human Exposure to Acoustic Energy", to participate in a subjective listening test. All subjects were given standard pure-tone audiogram hearing tests to determine their suitability for the study; all subjects had normal hearing. The subjects were not trained prior to the test.

This test was designed to answer the two questions fundamental to this work:

1. Does the use of a two-phase damped-exponential model in voiced-speech synthesis show improvement in quality over that obtained with one- and two-phase LPC or one-phase damped exponential models?
2. Does the use of a two-phase model in general show an improvement in quality over one-phase models?

To answer these questions, statistical analysis of the test results is needed. One analysis method well-suited for this task is the technique known as analysis of variance (ANOVA). With ANOVA in mind, a two factor, within-subjects listening test was designed. The two factors of interest are speaker variations and variations between analysis-synthesis methods (treatments).

Four speakers (fdrd1, fcmh0, mcmj0, mchh0) from the TIMIT data base form the 4 levels within the speaker factor. These speakers were chosen because there are

three common sentences uttered by each. Two females and two males were chosen to provide a good gender balance. The three sentences used are:

1. **sa1** "She had your dark suit in greasy wash water all year."
2. **sa2** "Don't ask me to carry an oily rag like that."
3. **sx284** "Jeff thought you argued in favor of a centrifuge purchase."

Four analysis-synthesis methods were applied to each TIMIT sentence-speaker combination. The four methods used include single-phase damped-exponential, single-phase LPC, two-phase damped-exponential, and two-phase LPC. Each method is performed pitch-synchronously as was described in Chapter III. The model orders used by the analysis-synthesis systems are 18 for each voiced phase and unvoiced frame. The value of 18 was chosen as a typical value for data sampled at a 16 kHz rate. Table 2 summarizes the speaker-treatment-sentence combinations for this test.

Table 2 Two factor within-subjects experiment block design (DE=Damped-Exponential).

	Treatment			
Speaker	1 phase DE	2 phase DE	1 phase LPC	2 phase LPC
fdrd1	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284
fcmh0	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284
mcmj0	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284
mchh0	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284	sa1 sa2 sx284

The listening test was conducted at Armstrong Laboratory at Wright-Patterson AFB, OH. The procedure followed for each subject (volunteer from the panel) was:

1. The subject entered the laboratory, was seated at a Sun SPARCstation 2, and given verbal instructions on the testing process. Instructions covered data entry using the keyboard and when to enter quality ratings.

2. The subject was presented with every speaker-treatment combination one block at a time. A block consisted of a set of three sentences from one speaker treated with one synthesis method. There were 16 blocks presented to the subject during the session. The subject was asked to rate the quality of each sentence on a scale from 1 (poor) to 5 (excellent). The order in which the blocks were presented was randomly determined using a uniform random number generator and was different for each subject. Within each block, the sentence order was also randomly determined.

3. At the beginning of each block presented to the subject, four "anchor" sentences were played. This "anchoring" was performed to provide lower and upper bounds of quality ratings to the subject prior to being asked to rate the quality of speech. These anchor sentences provided two examples (one male and one female) of speech that might be considered to have a quality of 1, and two examples (one male and one female) of speech that might be considered to have a quality of 5. The order of presentation was randomized between poor and excellent examples, and then again between male/female speakers within each quality boundary. The speakers and sentences chosen were different than those included in the rating portion: faks0 speaking sentence si1573 ("His captain was thin and haggard and his beautiful boots were worn and shabby."), and mjls0 speaking si1726 ("And men also used vacuum cleaners in both rooms, sucking dust up once more."). Single-phase, 10^{th} order, LPC was used to synthesize the poor quality examples while a single-phase, 50^{th} order, damped-exponential model was used for the excellent quality examples. These orders and models were identified empirically to have the quality levels desired for this listening test.

4. After the anchors were played, each of the three sentences to be rated was played in random order. The subject was asked to rate the overall quality of each of the three sentences.

5. After all 16 blocks were presented, the test was complete for that subject.

4.2 Listening Test Results

Figure 10 shows a plot of the quality scores averaged across all 19 subjects from each speaker and treatment. Given a quick look at the plots, one might draw the conclusion (though not completely accurate) that a two-phase damped-exponential model performed the best in general. However, because the mean quality scores for this treatment and one-phase damped-exponential model are nearly equal for speaker mcmj0, this conclusion cannot be stated as the true conclusion of the test. To draw well-substantiated claims from the data, a statistical analysis must be performed. A two factor within-subjects analysis of variance (ANOVA) is used to provide the analysis. However, this type of ANOVA is not the most appropriate method to use for ordinal data. Most parametric ANOVA methods assume a linear relationship of the data (scores) collected. The opinion scores collected in this listening test do not have this linear relationship; that is, an opinion score of (5) is not necessarily two times better than a (2), and a (5) is not necessarily five times better than a (1). Although it may not be the most appropriate test, it still provides insight into the relative performance of the four synthesis methods tested. An additional non-parametric test called the Friedman two-way ANOVA by ranks [29] is also conducted to add credence to the conclusions drawn from the parametric analysis.

4.2.1 Assumptions. There are four fundamental assumptions made when performing within-subjects ANOVA.

1. Homogeneity of within-factor variances (i.e. variances are equal).

2. Normally distributed factor populations.

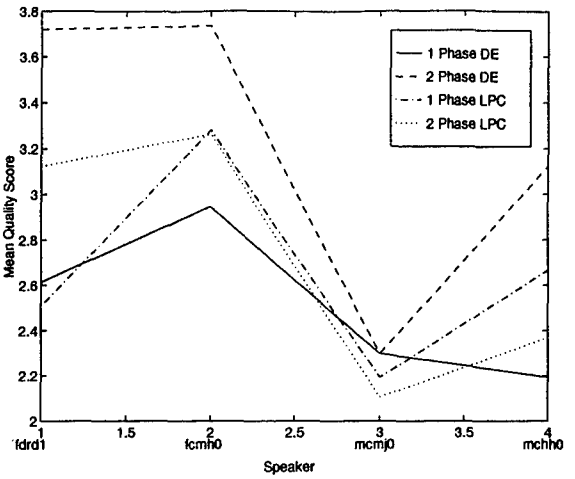

Figure 10 Quality scores averaged across subjects for each speaker and treatment.

3. Independence: The individual factor and between factor sum of squares "provide independent information about the outcome of the experiment" [13].

4. The sphericity assumption: The variances of differences between paired treatment scores are equal throughout the population.

The fourth assumption, sphericity, is frequently violated in human subjective tests [13]. The effect of this violation is to positively bias the intermediate result of the ANOVA. To correct for this positive bias, a very stringent correction is made to the critical F-value chosen. This correction, called the Geisser-Greenhouse correction, forces the critical F-value to be $f(1, n-1)$ for all F tests, where n is the number of subjects in the within-subjects ANOVA. This correction factor was used in all ANOVA tests performed in this work.

4.2.2 Other considerations. Certain phenomena occur when using human subjects to perform subjective listening tests. Two particular phenomena are the practice effect and effects of differential carryover.

The practice effect concerns the fact that a subject may show a general improvement in testing, or become bored or fatigued [13]. The listening test designed for this work attempted to minimize this effect by randomizing the presentation or-

der amongst the 19 subjects. Thus each block is presented at a different location within the test so that effects of training from one subject will be offset by the early presentation of that block to another subject.

The differential carryover effect concerns the effect an earlier presentation of a treatment has on the next presentation [13]. For example, presentation of a poor quality sample just prior to the presentation of a moderate quality sample may cause the rating of the moderate sample to be inflated. Presentation of a high quality sample prior to a moderate sample my deflate the rating of the moderate sample. The re-anchoring at the beginning of each block of this test was designed to minimize this effect.

4.2.3 Two Factor Within-Subjects ANOVA. With these assumptions and considerations in mind, the two factor within-subjects analysis of variance is performed as described in [13]. For this analysis, the mean score for each subject in each block of the listening test is used as an observation. As mentioned above, each factor has four levels of interest. Table 3 summarizes the ANOVA performed on the data collected in the subjective listening tests. Using the Geisser-Greenhouse correction and significance level of $\alpha = 0.05$, the critical F score is $f(1, n-1) = f(1, 18) = 4.41$.

Table 3 Summary of Analysis of Variance

Source	SS	df	MS	F
Speaker (Sp)	50.83	3	16.94	20.12
Treatment (T)	21.45	3	7.15	9.36
Sp × T	12.02	9	1.34	2.41
Subjects (S)	63.88	18	3.55	
Sp × S	45.48	54	0.84	
T × S	41.25	54	0.76	
Sp × T × S	89.92	162	0.56	

The first set of hypotheses to test is for interactions between the two factors (Speaker and Treatment). Then, if the null hypothesis for the between factors test is not rejected, the within factor interactions can be analyzed. Let the hypotheses

be:

$$
\begin{aligned}
H_0' &: \mu_T = \mu_{Sp} \text{ (i.e. between factors)} \\
H_a' &: \mu_T \neq \mu_{Sp} \\
H_0'' &: \mu_{fdrd1} = \mu_{fcmh0} = \mu_{mcmj0} = \mu_{mchh0} \text{ (i.e. between speakers)} \\
H_a'' &: \text{At least two of the } \mu_{speaker}\text{'s are different} \\
H_0''' &: \mu_{DE1} = \mu_{DE2} = \mu_{LPC1} = \mu_{LPC2} \text{ (i.e. between treatments)} \\
H_a''' &: \text{At least two of the } \mu_{treatment}\text{'s are different}
\end{aligned}
$$

where the $_0$ and $_a$ subscripts denote the null and alternate hypotheses respectively.

In Table 3, the row for Sp × T represents the summary of the between factor analysis. Since $f(Sp \times T) = 2.41 < 4.41$, the critical value, the null hypothesis H_0' is not rejected indicating no statistically significant (at an $\alpha = 0.05$ significance level) differences between the factor mean quality scores. Since this null hypothesis is not rejected,, we proceed to the within factor variations.

Since the f scores for each of the within factor tests exceed the critical value of 4.41, the corresponding null hypotheses (H_0'' and H_0''') are rejected. This rejection indicates there are statistically significant (again at the $\alpha = 0.05$ level) differences in the mean quality scores among speakers, and also among synthesis treatments.

4.2.4 Tukey test. Since the ANOVA showed statistically significant differences in the mean quality scores within factors, a Tukey test [13] is performed to analyze these differences. This test simply compares all pairwise differences in means within a given factor.

4.2.4.1 Within-Speaker Analysis. Table 4 summarizes the mean quality score differences for the speaker factor after performing the first step of the Tukey

test. The entries below the main diagonal of the table are omitted as they are mirror images of the entries above it.

Table 4 Pairwise differences of means for Speaker factor

Speaker	Speaker Means	mcmj0 2.2237	mchh0 2.5877	fdrd1 2.9912	fcmh0 3.3070
mcmj0	2.2237	—	0.3640	0.7675	1.0833
mchh0	2.5877		—	0.4035	0.7193
fdrd1	2.9912			—	0.3158
fcmh0	3.3070				—

The second step is to calculate the minimum pairwise mean difference that must be exceeded to show statistically significant differences. This threshold value is defined as [13]

$$\bar{d}_T = q_T \sqrt{\frac{MS_{S/A}}{n}} \tag{19}$$

where q_T is an entry in the table of the studentized range statistic, $MS_{S/A}$ is the error term from the overall ANOVA, and n is the number of samples for each level under analysis in the Tukey test.

For this work, $q_T = 3.76$ ($df_{error} = 54, k = 4$) as shown in Table A-5 in [13]. From Table 3, the error term is the MS for $SP \times S = 0.84$. Finally, the number of samples for each speaker, n, is 76 (19 subjects and 4 treatments per speaker). The resulting $\bar{d}_T$ is 0.3958.

Only the differences in means between each female speaker and each male speaker exceed the threshold value of 0.3958. Therefore, the conclusion drawn from this Tukey test is that speakers fdrd1 and fcmh0 had statistically significant (at the $\alpha = 0.05$ level) higher mean quality scores than both mcmj0 and mchh0 for the sentences and model orders chosen for this thesis.

4.2.4.2 Within-Treatment Analysis. Table 5 summarizes the mean quality score differences for the treatment factor.

Table 5 Pairwise differences of means for Treatment factor

Treatment	Treatment Means	DE1 2.5132	LPC1 2.6623	LPC2 2.7149	DE2 3.2193
DE1	2.5132	—	0.1491	0.2018	0.7061
LPC1	2.6623		—	0.0526	0.5570
LPC2	2.7149			—	0.5044
DE2	3.2193				—

The threshold value is calculated as in (19) with the same values for q_T and n as for the speaker comparisons, but the error term $MS_{S/A}$ is 0.76 versus 0.84 in the speaker comparisons. The resulting threshold value is $\overline{d}_T = 0.3770$.

Only the differences in means between the two-phase damped-exponential treatment (DE2) and all other treatments exceed the threshold value of 0.3770. Therefore, the conclusion drawn from this Tukey test is that the two-phase damped-exponential synthesis method resulted in statistically significant (at the $\alpha = 0.05$ level) higher mean quality scores than all other treatment methods used for the sentences, speakers, and model orders chosen for this thesis.

4.2.5 Non-Parametric Analysis Results. Analysis of the subjective listening test data using the Friedman two-way analysis of variance supports the conclusion that the two-phase damped-exponential model for speech synthesis performs better than the other synthesis methods tested. The Friedman test showed that the two-phase damped-exponential synthesis method was never outperformed by the other methods tested. This conclusion lends creedance to the conclusions drawn from the two-factor within-subjects ANOVA and Tukey tests performed above.

4.3 Summary

This chapter described the experiments and statistical analysis performed to test the analysis-synthesis systems described in Chapter III. It presented the subjective listening test performed at Armstrong Laboratory and the analysis of variance

and Tukey tests used to draw conclusions of statistical significance from the data collected. These tests indicate statistically significant (at the $\alpha = 0.05$ level) higher mean quality scores for females across all synthesis methods. Qualities of the male speakers such as higher pitch, softer voices, etc. may have contributed to this finding. The tests also show that the two-phase damped-exponential model resulted in statistically significant (at the $\alpha = 0.05$ level) mean quality scores across speakers than all other synthesis methods used. These results were reinforced using the non-parametric Friedman test. The next chapter will summarize these results and decide whether or not the goal of this thesis was achieved.

V. Conclusions and Recommendations

As mentioned in Chapter I, the purpose of this thesis is to answer the questions:

1. Does the use of a two-phase damped-exponential model in voiced-speech synthesis show improvement in quality over that obtained with one- and two-phase LPC or one-phase damped-exponential models?
2. Does the use of a two-phase model in general show an improvement in quality over one-phase models?

This chapter uses the results of the subjective listening test and statistical analyses presented in Chapter IV to determine the answers. Additionally, several areas in which this thesis could be applied for further study are discussed.

5.1 Conclusions

The ANOVA and Tukey tests of Chapter IV do show a statistically significant (at the $\alpha = 0.05$ level) increase in mean quality scores for the speech synthesized using the two-phase damped-exponential model for voiced speech over the other methods tested. However, since only four speakers, four treatment methods, and three sentences were used, more thorough tests are recommended to more conclusively address the primary question. The answer to the second question above is that the potential for quality improvement of a two-phase model in general was not shown to be significant in this particular test. Therefore, the conclusions are:

1. Yes, a two-phase damped-exponential model does show improvement in quality over some other synthesis methods in use today for the limited data set examined. Further testing is required.
2. No significant quality improvement could be shown for a two-phase model in general over one-phase models. Again, however, limited testing was done and perhaps further testing would show a significant improvement.

5.2 Potential areas for further research

With the demonstrated potential for improvement in quality of synthesis using a two-phase damped-exponential model for voiced speech comes the possibility to improve performance in a wide variety of speech applications. Before covering some of these applications, it must be noted that this speech model is inappropriate for any application where a very low bit-rate is desired. Each increase in model order in this method results in the addition of 2 parameters (complex pole and amplitude), each represented by a real and imaginary number. In addition, since two sets of parameters are estimated over each pitch period (analysis frame), there are twice the amount of parameters per frame as in conventional methods. Thus, the number of values required to represent each frame of speech increases by 8 for every increase in model order.

5.2.1 Speaker Identification. Speaker identification systems in use today incorporate a wide variety of classification techniques using numerous types of features. Some features commonly extracted from the speech for use with the classifier are mel frequency cepstral coefficients, linear prediction (LP) coefficients, LP cepstral coefficients, and transitional coefficients [19, 25]. Another set of parameters worthy of examination for possible classification improvement are the complex poles and amplitudes extracted in the two-phase damped-exponential analysis. While the increase in the number of parameters used to represent a frame of speech is detrimental in very low bit-rate coding applications, this increase could be beneficial in a speaker identification problem. Whether or not these parameters form a good feature set for speaker identification is not known leading to the need for further research. Furthermore, the discriminative power of features from within a pitch period has yet to be explored.

5.2.2 Speech Synthesis. This thesis has shown the potential to improve the quality of synthetic speech by using a two-phase damped-exponential model.

However, a significant improvement in quality was not deduced from the statistical analysis for a two-phase model in general. Perhaps further testing would show that statistically significant improvement or lead to improvements of the methods presented here. In addition, many of the subjects on the testing panel mentioned the presence of "noise" and other artifacts even though the fundamental speech quality was very good. Research into possible ways to remove this noise and the remaining artifacts may also improve the quality of synthetic speech. Another improvement to be made may be in the area of determining the phase transition point. Only one method was presented here. Improvements to this method could be made or a new method developed and used.

5.3 Thesis Summary

This work has presented evidence that the use of a two-phase damped-exponential model for voiced speech synthesis provided statistically significant (at the $\alpha = 0.05$ level) improvement in the quality of synthetic speech relative to single phase LPC, single phase damped exponential, and two-phase LPC techniques for the data set tested. Furthermore, this evidence was obtained using a new method for determining the phase transition point without the use of EGG data. The results of the tests accomplished are promising and show the need for further testing and research of this model to strengthen the conclusions and lead to improvements for the algorithms developed here. In addition, research into the application of this model to other areas of speech technology may give promising results.

Appendix A. Synthesizing With a Damped-Exponential Model

This appendix details the analysis-synthesis procedure for a single-phase damped-exponential model.

A.1 Synthesis Equation

The equation used to synthesize one segment of speech is:

$$s[n] = \sum_{i=0}^{R-1} A_i \rho_i^n, \quad n = 0, \ldots, N-1 \tag{20}$$

where N is the number of speech samples to synthesize, R is the number of complex poles estimated, A_i is the complex amplitude for the i^{th} pole, and ρ_i is the i^{th} complex pole.

Since each of the parameters are complex numbers in general, they can be written as:

$$A_i = B_i e^{j\phi_i}, \ \ and \tag{21}$$

$$\rho_i = C_i e^{j\omega_i} \tag{22}$$

where B_i, C_i, ϕ_i, and ω_i are real numbers. Substituting (21) and (22) into (20) yields:

$$s[n] = \sum_{i=0}^{R-1} B_i e^{j\phi_i} C_i^n e^{jn\omega_i} \tag{23}$$

or

$$s[n] = \sum_{i=0}^{R-1} Z_i e^{j(n\omega_i + \phi_i)}, \tag{24}$$

where $Z_i = B_i C_i^n$. Given that the poles estimated are real or in complex conjugate pairs, it becomes apparent from (24) that the synthetic waveform is simply a weighted sum of sinusoids which are exponentially damped with the damping controlled by the C_i^n term.

A.2 Pole Estimation

The complex poles are estimated using linear prediction in the backward time direction and singular value decomposition (SVD). The reason for the backward time analysis is that the waveform over one pitch period, when reversing the time indices, will appear to be unstable, and thus have unstable modes. Therefore, modes that are truly stable in the waveform will appear to be unstable due to the time reversal and the poles will have a magnitude greater than one. Extraneous poles appear inside the unit circle in the complex plane [17, 18, 24]. In practice, some poles may be estimated as slightly unstable and appear slightly inside the unit circle before reflection. Therefore, poles having a magnitude greater than a value selected by the user (typically 0.87-0.9 determined empirically) are retained and reflected about the unit circle. The next sections describe the estimation algorithm in more detail.

A.2.1 Linear Prediction Equations. The equation to solve is:

$$\mathbf{Ax} = -\mathbf{h} \tag{25}$$

where for L estimated poles and N samples in the analysis frame y:

$$\mathbf{x} = \begin{bmatrix} b(1) \\ b(2) \\ \vdots \\ b(L) \end{bmatrix} \tag{26}$$

and

$$[\mathbf{h}|\mathbf{A}] = \left[\begin{array}{c|cccc} y_0 & y_1 & y_2 & \cdots & y_L \\ y_1 & y_2 & y_3 & \cdots & y_{L+1} \\ y_2 & y_3 & y_4 & \cdots & y_{L+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{N-L-1} & y_{N-L} & y_{N-L+1} & \cdots & y_{N-1} \end{array}\right] \tag{27}$$

Equations (25-27) show that the each data point is modeled as a weighted sum of the future L samples.

The least squares solution of (25) for $\mathbf{x}$ is

$$\mathbf{x} = -\mathbf{A}^{\dagger}\mathbf{h} \tag{28}$$

This form of the equation is the one we will use to estimate the poles in MATLAB®. The vector $\mathbf{x}$ contains the coefficients for the prediction polynomial [18]

$$B(z) = 1 + b(1)z^{-1} + b(2)z^{-2} + \cdots + b(L)z^{-L} \tag{29}$$

from which the complex poles are determined.

A.2.2 Solving for $\mathbf{x}$. Typically there are fewer true filter poles (M) than the number estimated (L), (i.e. $L > M$). To alleviate ill-conditioning of $\mathbf{A}$, the SVD of $[\mathbf{h}|\mathbf{A}]$ is used [18]. During this process, small singular values ($< 10^{-9}$) are set to zero effectively increasing the SNR of the data prior to solving for $\mathbf{x}$.

Using the SVD, $\mathbf{A}$ can be represented as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{H} \tag{30}$$

where $\mathbf{U}$ is a $(N-L) \times (N-L)$ unitary matrix of the "left" singular vectors, $\mathbf{\Sigma}$ is a $(N-L) \times L$ matrix of nonnegative real singular values, and $\mathbf{V}^{H}$ is the hermitian (conjugate transpose) of the $L \times L$ matrix of "right" singular vectors [31]. As mentioned previously, the small real singular values in $\mathbf{\Sigma}$ are set to zero. Thus,

instead of using $\mathbf{\Sigma}$:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_L \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{(N-L)\times L} \tag{31}$$

we use

$$\hat{\mathbf{\Sigma}} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & \sigma_2 & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \sigma_R & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{(N-L)\times L} \tag{32}$$

where $R \leq L$. Substitution into (25) yields:

$$\mathbf{U}\hat{\mathbf{\Sigma}}\mathbf{V}^H\mathbf{x} = -\mathbf{h} \tag{33}$$

Solving for $\mathbf{x}$ yields:

$$\hat{\mathbf{x}} = -\mathbf{V}\hat{\mathbf{\Sigma}}^{\dagger}\mathbf{U}^H\mathbf{h} \tag{34}$$

where

$$\hat{\Sigma}^{\dagger} = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_R} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{(N-L)\times L} \tag{35}$$

A.2.3 $\boldsymbol{\rho}$ *determination.* Now that $\mathbf{x}$ is known (i.e. the coefficients of the prediction polynomial), we can use MATLAB®'s "roots" function to determine the complex vector of estimated true and extraneous poles, $\boldsymbol{\theta}$. The magnitude of each complex pole is examined to identify those poles that are to be retained. If the magnitude of the i^{th} element of $\boldsymbol{\theta}$ is less than the threshold, the pole is rejected. If the magnitude is greater than the user selected threshold, the conjugate-reciprocal ($\frac{1}{\theta_i^*}$) is retained. This process reflects the pole about the unit circle in the z-plane. The resultant vector, $\boldsymbol{\rho}$, contains each of the estimated true poles for the vocal tract filter.

A.3 Determination of A_i's in synthesis equation

Next we need to determine how much each pole contributes to the waveform. The A_i's in (20) represent these contributions. To estimate them, we use a matrix form of (20) with the original data sequence y replacing the estimated sequence $\hat{s}(n)$:

$$\mathbf{y} = \mathbf{\Lambda a} \tag{36}$$

or

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \rho_0 & \rho_1 & \cdots & \rho_{R-1} \\ \rho_0^2 & \rho_1^2 & \cdots & \rho_{R-1}^2 \\ \vdots & \vdots & & \vdots \\ \rho_0^{N-1} & \rho_1^{N-1} & \cdots & \rho_{R-1}^{N-1} \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_{R-1} \end{bmatrix} \tag{37}$$

Solving for **a** yields:

$$\mathbf{a} = \mathbf{\Lambda}^{\dagger}\mathbf{y} \tag{38}$$

which is easily solvable in MATLAB® using the "pinv" routine.

Now that the A_i's and ρ_i's from (20) have been determined, the new frame can be synthesized.

A.4 An Example

As an example, let's use the frame of data illustrated in Figure 11(a). This data is one pitch period extracted from the phoneme /iy/ (as in "she") in the sentence "sa1" uttered by speaker "mefg0" in the TIMIT data base. The model parameters are $L = 18$, $N = 145$, and $threshold = 0.87$.

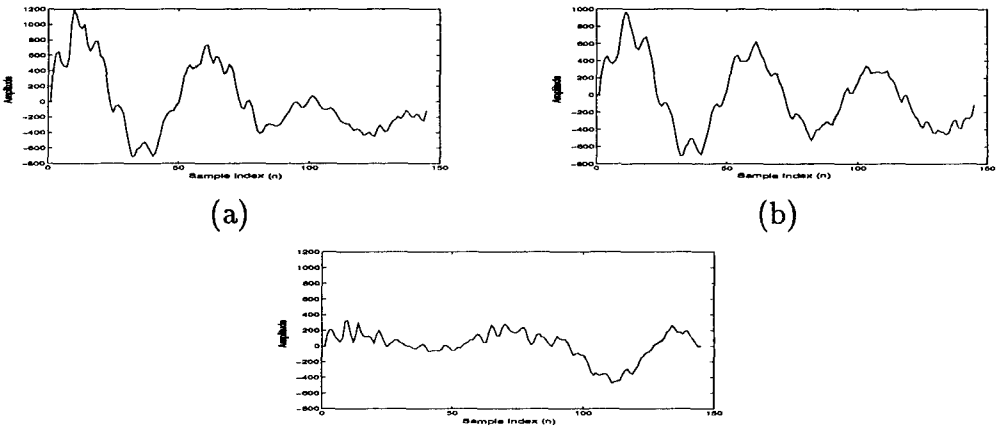

Figure 11 (a) One pitch period from the phoneme /iy/ as in "she". (b) Re-synthesized waveform. (c) Plot of the error between the two.

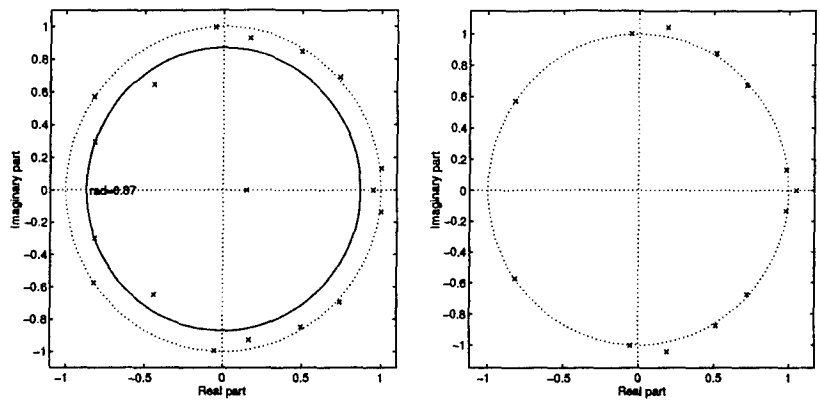

Figure 12 (a): The estimated true and extraneous poles ($\boldsymbol{\theta}$), (b): The true poles retained and reflected about the unit circle ($\boldsymbol{\rho}$).

Using these parameters and data, the resulting $\boldsymbol{\theta}$ is as shown in Table 6. Figure 12(a) shows the location of these poles on the z-plane with a solid ring designating the 0.87 threshold. It can be seen from the magnitudes that only 13 elements with a magnitude greater than 0.87 will be retained. The remaining five will be rejected.

Reflecting the first 13 poles about the unit circle yields the estimated pole vector, ρ, shown in the second column of Table 6 and illustrated in Figure 12(b).

Using $\boldsymbol{\rho}$ and the original data vector, the complex amplitude vector, $\mathbf{a}$, (shown in the right hand column of Table 6) is estimated. Examination of the elemental magnitudes of this vector illustrates an important feature of this analysis-synthesis method. For this example, the significant modes (a mode being a pole/amplitude pair) are those with complex magnitudes greater than or equal to 1.8441. These modes correspond to the true modes as illustrated by the LPC spectra in Figure 13. Those modes with smaller complex amplitudes correspond to a DC offset and an extraneous mode; each of these modes would be rejected by extending the radius threshold to 0.96. However, because they are kept, their impact on the synthesized waveform is minimized by their small complex amplitudes in $\mathbf{a}$. These modes are not

Table 6 Estimated parameters. Frequency figures are based on a 16 kHz sample rate.

θ		ρ		freq (Hz)	a	
0.1513 ∠	0.0000°					
0.9513 ∠	0.0000°	1.0512 ∠	0.0000°	0	0.2244 ∠	180.0000°
1.0088 ∠	7.6475°	0.9912 ∠	7.6475°	340	434.6135 ∠	91.2425°
1.0088 ∠	−7.6475°	0.9912 ∠	−7.6475°	-340	434.6135 ∠	−91.2425°
1.0140 ∠	43.1274°	0.9862 ∠	43.1274°	1916.8	109.0635 ∠	80.3428°
1.0140 ∠	−43.1274°	0.9862 ∠	−43.1274°	-1916.8	109.0635 ∠	−80.3428°
0.9834 ∠	59.5625°	1.0169 ∠	59.5625°	2647.2	1.8441 ∠	−123.4215°
0.9834 ∠	−59.5625°	1.0169 ∠	−59.5625°	-2647.2	1.8441 ∠	123.4215°
0.9424 ∠	79.7025°	1.0611 ∠	79.7025°	3542.3	0.0073 ∠	8.4519°
0.9424 ∠	−79.7025°	1.0611 ∠	−79.7025°	-3542.3	0.0073 ∠	−8.4519°
0.9969 ∠	93.0193°	1.0031 ∠	93.0193°	4134.2	8.1502 ∠	153.4833°
0.9969 ∠	−93.0193°	1.0031 ∠	−93.0193°	-4134.2	8.1502 ∠	−153.4833°
0.7818 ∠	124.0899°					
0.7818 ∠	−124.0899°					
1.0021 ∠	144.9657°	0.9979 ∠	144.9657°	6442.9	6.3134 ∠	96.8941°
1.0021 ∠	−144.9657°	0.9979 ∠	−144.9657°	-6442.9	6.3134 ∠	−96.8941°
0.8689 ∠	159.9182°					
0.8689 ∠	−159.9182°					

apparent in the LPC spectrum. Solving (20) results in the synthetic waveform shown in Figure 11(b). The error between the two waveforms is illustrated in Figure 11(c).

A.4.1 Pole Rejection Analysis. If we were to reject the poles at 3.542 kHz and DC, what would happen? Figure 14 shows (a) the original waveform, (b) the reconstructed waveform including those modes, (c) the reconstructed waveform excluding those modes, and (d) the error between the two synthetic waveforms.

Figure 15 shows the new LPC spectrum overlayed on the previous two. With the absence of the 3.542 kHz mode, there is a sharper dip in the spectrum at this point, causing an ill-match of the spectrum. So even though the mode wasn't apparent in the spectrum containing all the poles, it was still an important component

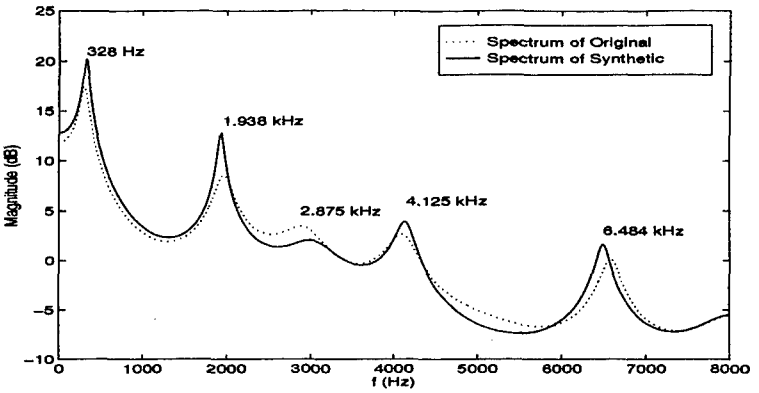

Figure 13 Plot of the LPC spectrum for the original and synthetic waveforms.

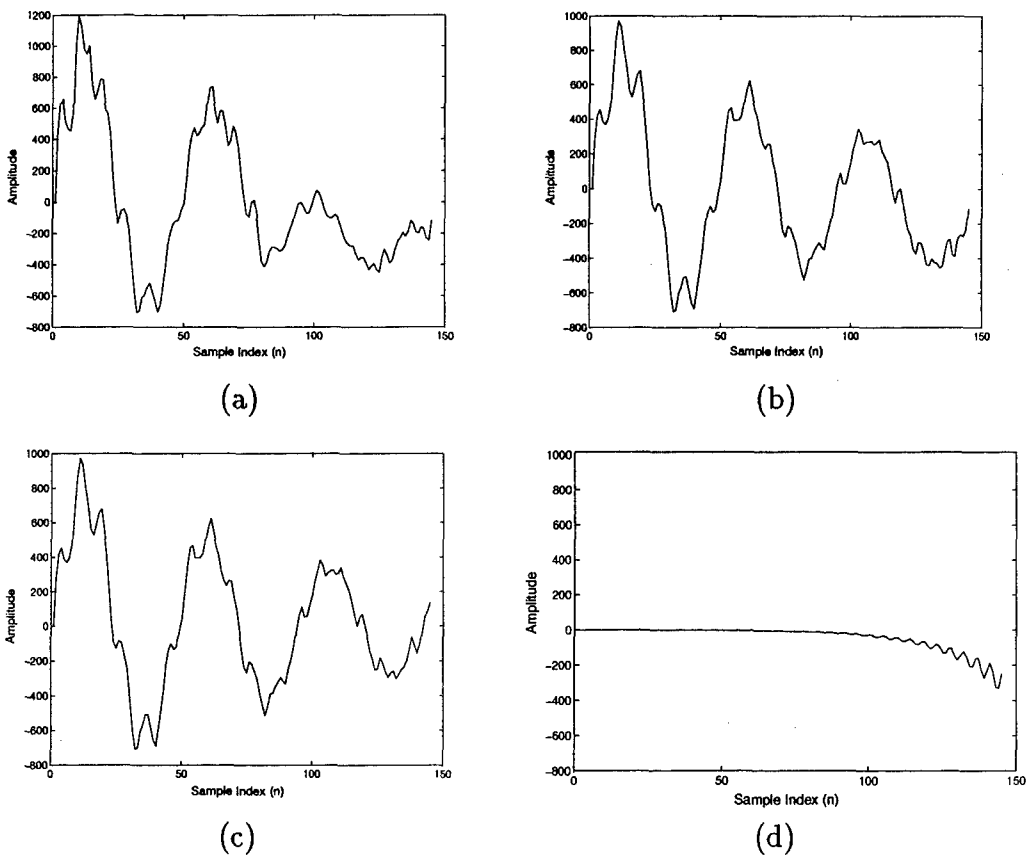

Figure 14 (a) Original waveform. (b) Re-synthesized waveform including DC and 3.542 kHz modes. (c) Plot of the reconstructed waveform excluding the two modes, (d) The error between the two synthetic waveforms of (b) and (c).

from a spectral perspective. However, the error between the two synthetic waveforms shown in Figure 14(d) is nearly zero over much of the frame. Since it is nearly impossible to make any quality assessments on hearing a single pitch period of speech, the impact of rejecting the mode cannot be empirically determined from a single pitch period of speech through a listening test.

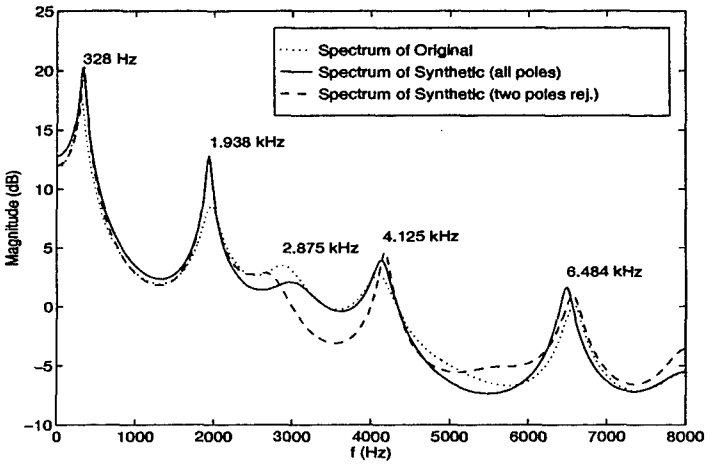

Figure 15 Plot of the LPC spectrum for the original and synthetic waveforms including the spectrum without poles at DC and 3.542 kHz.

Appendix B. MATLAB® *Code and Support Files*

For more information or to obtain copies of any or all code, e-mail: harb@afit.af.mil or mdesimio@afit.af.mil.

B.1 Single-Phase Damped-Exponential Master Routine

```
% function [y,origpoles,savepoles,numv,numuv]=onephasedex(ifile,pfile,...
%  phnfile,singord,uvord,fs,radius,maxper,uvpitch)
%
% Function: onephasedex.m
% Description: This function analyzes the incoming data ("data") using
% exponentially damped sinusoids as the model for the speech waveform.
% It performs sinusoid frequency (pole) analysis and complex damping
% factor analysis on a pitch-synchronous level.  The synthetic waveform
% is the sum of the exponentially damped sinusoids.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 30 Jul 96 -- Added comments, pre-emphasis for unvoiced frames,
%                        removed "energy" stuff.
%            2 Aug 96 -- Added initial conditions for unvoiced frames.
%            6 Aug 96 -- Modified code to read in binary data vice having it
%                       passed as an input parameter.
%           12 Aug 96 -- Modified function header to return the original
%                        and saved poles as well as the number of voiced
%                        and unvoiced frames.
%
% Input parameters:
%     ifile: The original speech signal data file name.
%     pfile: The name of the binary data file containing the pitch marks.
%      Each pitch mark should be some non-zero number.
%     phnfile: The name of the hand-labelled phoneme file for the speaker and
%          sentence corresponding to "data".
%     singord: The order for the pole determination analysis to be performed
%              over the voiced portion of "data".  This number corresponds
%              to the number of complex poles estimated.  If this is an even
%              number, there will be singord/2 complex conjugate pairs of
%              poles.
%     uvord: The order for the LPC analysis to be performed over the unvoiced
%            portion of "data".
%     fs: The sampling frequency.
%     radius: The minimum radius on the z-plane for accepting poles predicted
%             using the "backward" linear prediction method.
%     maxper: The maximum allowable period for voiced speech.  Used in
```

```
%             determining pitch marks when there are large gaps.
%     uvpitch: The desired frame rate during long periods (> maxper) of
%          unvoiced speech.
%
% Output Parameter:
%     y: The synthetic waveform.
% origpoles: A matrix whos columns contain the complex poles as estimated
%            using the backward prediction process for voiced frames.
%            This matrix contains all 'order' poles prior to selection/
%            reflection.
% savepoles: A matrix whos columns contain the selected and reflected
%            complex poles estimated using the backward prediction process
%            for voiced frames.
%   numv: The number of voiced frames.
%  numuv: The number of unvoiced frames.
%
% Subroutines directly called:
%     read_dat.m
%     newpmrks.m
%     readphn.m
%     loadphndb.m
%     synthlpc.m
%     synthv.m
%     detvoice.m
%
% Subroutines indirectly called:
%     covlpc.m
%     getCoef.m
%     getAmps.m
%     reconst.m
%     diffeq.m
%
function [y,origpoles,savepoles,numv,numuv]=onephasedex(ifile,pfile,phnfile...
      ,singord,uvord,fs,radius,maxper,uvpitch)

%
% read in data and pitch marks
%
data=read_dat(ifile,'short');
pmarks=read_dat(pfile,'short');

% re-calculate pitch marks back to a zero-crossing

newmarks=newpmrks(data,pmarks,fs,maxper,uvpitch);
```

```
%
% initialize some variables
%
numframes=length(newmarks(:,1));
y=[];
maxord=max([singord;uvord]);
origpoles=zeros(maxord(1),1);
savepoles=origpoles;
numuv=0;
numv=0;

%
% read in phoneme file
%

[phnind,phnval]=readphn(phnfile);
phndb=loadphndb;

%
% Loop through each pitch period generated with epochs and "newpmarks"
%
for i=1:numframes,

    samples=data(newmarks(i,1):newmarks(i,2));
%
% identify which phoneme we're in
%
    cur_phoneme=find((phnind(:,1)<=newmarks(i,1))&(phnind(:,2)>=newmarks(i,1)));
%
% Make crude voicing determination using phoneme file.
%
   vuv=detvoice(phnval(cur_phoneme,1:4),phndb);
   if vuv
      numv=numv+1;
%
% Synthesize new frame.
%
      [newframe,orig,sv]=synthv(samples,singord,radius);
%
% Save the poles originally estimated and those retained after
% selection/reflection.
%
      origpoles=[origpoles [orig;zeros(length(origpoles(:,1))-length(orig),1)]];
      savepoles=[savepoles [sv;zeros(length(savepoles(:,1))-length(sv),1)]];
%
```

```
% Add new frame to synthetic vector
%
      y=[y;newframe];

    else
%
% Unvoiced portion of the code.
%
    numuv=numuv+1;
%
% Synthesize unvoiced frame.  There is pre-emphasis.  Initial conditions are
% assumed to be zero as it is less important for unvoiced segments.
%
% Get past p=singord values of synthesized speech for use in LPC analysis.  If
% we are on the first frame, assume initial conditions are zero.
%
      if i==1,
         past=zeros(uvord,1);
      else
 past=y(length(y)-uvord+1:length(y));
      end;
      newframe=synthlpc(samples,uvord,past,0);

%
% Add on new frame
%
y=[y;newframe];
    end;
end; % for

save onephasedex
```

B.2 Two-Phase Damped-Exponential Master Routine

```
% function [y,phspnt,origpoles,savepoles,numv,numuv]=twophasedex(ifile,pfile,...
%          phnfile,iniord,secord,uvord,fs,radius,maxper,uvpitch)
%
% Function: twophasedex.m
% Description: This function analyzes the incoming data ("data") using
% exponentially damped sinusoids as the model for the speech waveform.
% It performs sinusoid frequency (pole) analysis and complex damping factor
% analysis on a pitch-synchronous level.  However, each voiced pitch period
% is subdivided into two frames.  The transition point is determined by
% estimating the model parameters over a short frame at the beginning of the
% pitch period, a short frame at the end, and re-synthesizing the frame using
```

```
% each set of model parameters.  Using the error between the two synthetic
% pitch periods and the original, the transition point is identified as the
% point of intersection of the two error waveforms. The synthetic waveform
% is the sum of the exponentially damped sinusoids characterized by model
% parameters estimated over each new subframe.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 31 Jul 96 -- Added comments, pre-emphasis for unvoiced frames,
%                        removed "energy" stuff.
%            2 Aug 96 -- Added initial conditions for unvoiced LPC synth.
%            6 Aug 96 -- Modified code to read in binary data vice having it
%                        passed as an input parameter.
%           12 Aug 96 -- Modified function header to return the original
%                        and saved poles as well as the number of voiced
%                        and unvoiced frames.
%
% Input parameters:
%     ifile: The original speech signal data file name.
%     pfile: The name of the binary data file containing the pitch marks.
%      Each pitch mark should be some non-zero number.
%     phnfile: The name of the hand-labeled phoneme file for the speaker and
%          sentence corresponding to "data".
%     iniord: The order for the pole determination analysis to be performed
%             over the initial frame of each voiced pitch period of "data".
%             This number corresponds to the number of complex poles
%             estimated.  If this is an even number, there will be iniord/2
%   complex conjugate pairs of poles.
%     secord: The order for the pole determination analysis to be performed
%             over the second frame of each voiced pitch period of "data".
%     uvord: The order for the LPC analysis to be performed over the unvoiced
%            portion of "data".
%     fs: The sampling frequency.
%     radius: The minimum radius on the z-plane for accepting poles predicted
%             using the "backward" linear prediction method.
%     maxper: The maximum allowable period for voiced speech.  Used in
%             determining pitch marks when there are large gaps.
%     uvpitch: The desired frame rate during long periods (> maxper) of
%          unvoiced speech.
%
% Output Parameter:
%      y: The synthetic waveform.
% phspnt: A vector containing each pitch marker denoted by a +1 and frame
%         transition points denoted by a -1.
% origpoles: A matrix whos columns contain the complex poles as estimated
```

```
%            using the backward prediction process for voiced frames.
%            This matrix contains all 'order' poles prior to selection/
%            reflection.
% savepoles: A matrix whos columns contain the selected and reflected
%            complex poles estimated using the backward prediction process
%            for voiced frames.
%   numv: The number of voiced frames.
%  numuv: The number of unvoiced frames.
%
% Subroutines directly called:
%     read_dat.m
%     newpmrks.m
%     readphn.m
%     loadphndb.m
%     synthlpc.m
%     synthv.m
%     detvoice.m
%     calctrans.m
%
% Subroutines indirectly called:
%     covlpc.m
%     getCoef.m
%     getAmps.m
%     reconst.m
%     calcerror.m
%     synth4tran.m
%     diffeq.m
%

function [y,phspnt,origpoles,savepoles,numv,numuv]=twophasedex(ifile,pfile,...
        phnfile,iniord,secord,uvord,fs,radius,maxper,uvpitch)

%
% read in data and pitch marks
%
data=read_dat(ifile,'short');
pmarks=read_dat(pfile,'short');

% re-calculate pitch marks back to a zero-crossing

newmarks=newpmrks(data,pmarks,fs,maxper,uvpitch);
%
% initialize some variables
%
numframes=length(newmarks(:,1));
```

```
y=[];
maxord=max([iniord,secord,uvord]);
origpoles=zeros(maxord(1),1);
savepoles=origpoles;
numphase=[];
numuv=0;
numv=0;
num2phs=0;
%
% read in phoneme file
%
[phnind,phnval]=readphn(phnfile);
phndb=loadphndb;
phspnt=[];
tpoints=zeros(length(data),1);
%
% Loop through each pitch period generated with epochs and "newpmarks"
%
for i=1:numframes,

    samples=data(newmarks(i,1):newmarks(i,2));
%
% identify which phoneme we're in
%
    cur_phoneme=find((phnind(:,1)<=newmarks(i,1))&(phnind(:,2)>=newmarks(i,1)));
    tpoints(newmarks(i,1))=1;
%
% Make crude voicing determination using phoneme file.
%
   vuv=detvoice(phnval(cur_phoneme,1:4),phndb);

   if vuv

%
% Calculate Phase Transition Point
%
numv=numv+1;
      tranpoint=calctrans(samples,iniord,secord,radius);
      phspnt=[phspnt tranpoint];
%
% Synthesize from sample 1 to the transition point and then from the
% transition point+1 to the end of the period.
%
num2phs=num2phs+1;
tpoints(newmarks(i,1)+tranpoint-1)=-1;
```

```
%
% Check to see if the order is greater than the number of sample in initial
% frame.  If so, move the transition point.
%
      if tranpoint<=iniord,
   tranpoint=iniord+5;
end;
%
% Synthesize the new initial frame and save poles.
%
      [newframe,og,sv]=synthv(samples(1:tranpoint),iniord,radius);
      origpoles=[origpoles [og;zeros(length(origpoles(:,1))-length(og),1)]];
      savepoles=[savepoles [sv;zeros(length(savepoles(:,1))-length(sv),1)]];
      numphase=[numphase 2];

%
% Check to see if order is greater than the number of points in the second
% frame.  If so, move the transition point back.
%
if (length(samples)-tranpoint) <= secord,
   tranpoint=length(samples)-secord-5;
      end;
%
% Synthesize the second frame and save poles.
%
      [temp,og,sv]=synthv(samples(tranpoint+1:length(samples)),secord,radius);
      origpoles=[origpoles [og;zeros(length(origpoles(:,1))-length(og),1)]];
      savepoles=[savepoles [sv;zeros(length(savepoles(:,1))-length(sv),1)]];
      numphase=[numphase 2];
%
% Check to see if there was a shift in transition point for the second frame.
% An indication will be too many points in the resulting two subframes.  If so
% Overlap and average the first extra points of the second frame with the last
% extra points of the first frame.
%
      if (length(temp)+length(newframe)) > length(samples)
         diff=length(temp)+length(newframe)-length(samples);
         tmpnf=newframe(length(newframe)-diff);
         tmpnf=[tmpnf;(newframe(length(newframe)-diff+1:length(newframe))...
  +temp(1:diff))./2;temp(diff+1:length(temp))];
   newframe=tmpnf;
      else
         newframe=[newframe;temp];
      end;
%
```

```
% Add new pitchperiod onto synthetic vector.
%
      y=[y;newframe];
   else
%
% Unvoiced portion of the code.  Covariance method LPC analysis is performed
% to determine vocal tract filter coefficients.  These then define a fileter
% which is excited by unit energy white noise.  Nno phase transition point
% is calculated and a single phase is assumed.
%
      numuv=numuv+1;
      if i==1,
         past=zeros(uvord,1);
      else
         past=y(length(y)-uvord+1:length(y));
      end;
      newframe=synthlpc(samples,uvord,past,0);
      numphase=[numphase 1];
%
% Add new frame to synthetic vector.
%
      y=[y;newframe];
  end;
end; % for
save twophasedex
```

B.3 Single-Phase LPC Master Routine

```
% function y=onephaselpc(ifile,pfile,phnfile,singord,uvord,fs,maxper,uvpitch)
%
% Function: onephaselpc.m
% Description: This function analyzes the incoming data ("data") using
% LPC models.  It then uses the LPC coefficients and gain to re-synthesize
% the waveform using the coefficients and gain to describe the reconstruction
% filter, and drives it with an impulse for voiced speech, and unit variance,
% zero mean white noise for unvoced.  This function performs pitch-synchr-
% onous analysis-synthesis.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 30 Jul 96 -- Added pre-emphasis for unvoiced frames.
%       -- Removed all "energy" tweaking.
%            2 Aug 96 -- Removed pre-emphasis.
%            6 Aug 96 -- Modified code to read in binary data vice having it
%                        passed as an input parameter.
```

```
%
% Input parameters:
%     ifile: The original speech signal data file name.
%     pfile: The name of the binary data file containing the pitch marks.
%      Each pitch mark should be some non-zero number.
%     phnfile: The name of the hand-labelled phoneme file for the speaker and
%          sentence corresponding to "data".
%     singord: The order for the LPC analysis to be performed over the voiced
%    portion of "data".
%     uvord: The order for the LPC analysis to be performed over the unvoiced
%  portion of "data".
%     fs: The sampling frequency.
%     maxper: The maximum allowable period for voiced speech.  Used in
%             determining pitch marks when there are large gaps.
%     uvpitch: The desired frame rate during long periods (> maxper) of
%          unvoiced speech.
%
% Output Parameter:
%     y: The synthetic waveform.
%
% Subroutines directly called:
%     read_dat.m
%     newpmrks.m
%     readphn.m
%     loadphndb.m
%     synthlpc.m
%     detvoice.m
%
% Subroutines indirectly called:
%     covlpc.m
%     diffeq.m
function y=onephaselpc(ifile,pfile,phnfile,singord,uvord,fs,maxper,uvpitch)

%
% read in pitch marks
%
data=read_dat(ifile,'short');
pmarks=read_dat(pfile,'short');

% re-calculate pitch marks back to a zero-crossing as well as break up
% long periods between marks identifying some unvoiced regions.

newmarks=newpmrks(data,pmarks,fs,maxper,uvpitch);

%
```

```
% initialize some variables
%
numframes=length(newmarks(:,1));
y=[];
numuv=0;
numv=0;
%
% read in phoneme file
%
[phnind,phnval]=readphn(phnfile);
phndb=loadphndb;
%
% Loop through each pitch period generated with epochs and "newpmarks"
%
for i=1:numframes,
    samples=data(newmarks(i,1):newmarks(i,2));
%
% identify which phoneme we're in
%
    cur_phoneme=find((phnind(:,1)<=newmarks(i,1))&(phnind(:,2)>=newmarks(i,1)));
%
% Make crude voicing determination using phoneme file.
%
   vuv=detvoice(phnval(cur_phoneme,1:4),phndb);
   if vuv
numv=numv+1;
%
% Get past p=singord values of synthesized speech for use in LPC analysis.  If
% we are on the first frame, assume initial conditions are zero.
%
      if i==1,
         past=zeros(singord,1);
      else
   past=y(length(y)-singord+1:length(y));
      end;
%
% Analyze and synthesize the frame
%
      newframe=synthlpc(samples,singord,past,1);
%
% Add new frame onto synthesized waveform
%
      y=[y;newframe];
   else
%
```

```
% Unvoiced portion of the code.
%
numuv=numuv+1;
%
% Synthesize unvoiced frame.  There is pre-emphasis.  Initial conditions are
% assumed to be zero as it is less important for unvoiced segments.
%
%
% Get past p=singord values of synthesized speech for use in LPC analysis.  If
% we are on the first frame, assume initial conditions are zero.
%
      if i==1,
         past=zeros(uvord,1);
      else
 past=y(length(y)-uvord+1:length(y));
      end;
      newframe=synthlpc(samples,uvord,past,0);
%
% Add on new frame
%
      y=[y;newframe];
   end;
end; % for

save onephaselpc
```

B.4 Two-Phase LPC Master Routine

```
% function [y,phspnt]=twophaselpc(ifile,pfile,phnfile,iniord,secord,uvord,fs,...
%          radius,maxper,uvpitch)
%
% Function: twophaselpc.m
% Description: This function analyzes the incoming data ("data") using
% LPC models.  Each voiced pitch period is subdivided into two frames.  The
% transition point is determined by estimating the model parameters over a
% short frame at the beginning of the pitch period, a short frame at the end,
% and re-synthesizing the frame using each set of model parameters.  Using
% the error between the two synthetic pitch periods and the original, the
% transition point is identified as the point of intersection of the two
% error waveforms. It then uses the covariance method LPC to re-synthesize
% the waveform using the coefficients and gain to describe the reconstruction
% filter, and drives it with an impulse for voiced speech, and unit energy,
% zero mean white noise for unvoiced.The synthetic waveform is computed over
% each subframe for voiced speech.
%
```

```
% Author: Capt Al Arb, USAF
% Date:  2 Aug 96
% Modified: 6 Aug 96 -- Modified code to read in binary data vice having it
%                       passed as an input parameter.
%
% Input parameters:
%     ifile: The original speech signal data file name.
%     pfile: The name of the binary data file containing the pitch marks.
%      Each pitch mark should be some non-zero number.
%     phnfile: The name of the hand-labeled phoneme file for the speaker and
%          sentence corresponding to "data".
%     iniord: The order for the LPC analysis to be performed over the initial
%             phase of the voiced portion of "data".
%     secord: The order for the LPC analysis to be performed over the second
%             phase of the voiced portion of "data".
%     uvord: The order for the LPC analysis to be performed over the unvoiced
%            portion of "data".
%     fs: The sampling frequency.
%     radius: The minimum radius on the z-plane for accepting poles predicted
%             using the "backward" linear prediction method.  Used in
%             transition point determiniation only.
%     maxper: The maximum allowable period for voiced speech.  Used in
%             determining pitch marks when there are large gaps.
%     uvpitch: The desired frame rate during long periods (> maxper) of
%          unvoiced speech.
%
% Output Parameter:
%      y: The synthetic waveform.
% phspnt: A vector containing each pitch marker denoted by a +1 and frame
%         transition points denoted by a -1.
%
% Subroutines directly called:
%     read_dat.m
%     newpmrks.m
%     readphn.m
%     loadphndb.m
%     synthlpc2.m
%     detvoice.m
%     calctrans.m
%
% Subroutines indirectly called:
%     covlpc.m
%     getCoef.m
%     getAmps.m
%     reconst.m
```

```
%     calcerror.m
%     synth4tran.m
%     diffeq.m
%

function [y,phspnt]=twophaselpc(ifile,pfile,phnfile,iniord,secord,uvord,fs,...
         radius,maxper,uvpitch)

%
% read in data and pitch marks
%
data=read_dat(ifile,'short');
pmarks=read_dat(pfile,'short');

% re-calculate pitch marks back to a zero-crossing

newmarks=newpmrks(data,pmarks,fs,maxper,uvpitch);
%
% initialize some variables
%
numframes=length(newmarks(:,1));
y=[];
maxord=max([iniord,secord,uvord]);
numphase=[];
numuv=0;
numv=0;
num2phs=0;
%
% read in phoneme file
%
[phnind,phnval]=readphn(phnfile);
phndb=loadphndb;
phspnt=[];
tpoints=zeros(length(data),1);
%
% Loop through each pitch period generated with epochs and "newpmarks"
%
for i=1:numframes,

    samples=data(newmarks(i,1):newmarks(i,2));
%
% identify which phoneme we're in
%
    cur_phoneme=find((phnind(:,1)<=newmarks(i,1))&(phnind(:,2)>=newmarks(i,1)));
    tpoints(newmarks(i,1))=1;
```

```
%
% Make crude voicing determination using phoneme file.
%
   vuv=detvoice(phnval(cur_phoneme,1:4),phndb);

   if vuv

%
% Calculate Phase Transition Point
%
      numv=numv+1;
      tranpoint=calctrans(samples,iniord,secord,radius);
      phspnt=[phspnt tranpoint];
%
% Synthesize from sample 1 to the transition point and then from the
% transition point+1 to the end of the period.
%
num2phs=num2phs+1;
tpoints(newmarks(i,1)+tranpoint-1)=-1;
%
% Check to see if the order is greater than the number of sample in initial
% frame.  If so, move the transition point.
%
      if tranpoint<=iniord,
   tranpoint=iniord+5;
      end;
%
% Get past p=iniord values of synthesized speech for use in LPC analysis.  If
% we are on the first frame, assume initial conditions are zero.
%
      if i==1,
         past=zeros(iniord,1);
      else
   past=y(length(y)-iniord+1:length(y));
      end;
%
% Synthesize the new initial frame.
%
      newframe=synthlpc2(samples(1:tranpoint),iniord,past,1,1);
      numphase=[numphase 2];

%
% Check to see if order is greater than the number of points in the second
% frame.  If so, move the transition point back.
%
```

```
if (length(samples)-tranpoint) <= secord,
   tranpoint=length(samples)-secord-5;
        end;
%
% Get past p=secord values of synthesized speech for use in LPC analysis.
%
past=newframe(length(newframe)-secord+1:length(newframe));

%
% Synthesize the second frame.
%
      temp=synthlpc2(samples(tranpoint+1:length(samples)),secord,past,1,2);
      numphase=[numphase 2];
%
% Check to see if there was a shift in transition point for the second frame.
% An indication will be too many points in the resulting two subframes.  If so
% Overlap and average the first extra points of the second frame with the last
% extra points of the first frame.
%
      if (length(temp)+length(newframe)) > length(samples)
         diff=length(temp)+length(newframe)-length(samples);
         tmpnf=newframe(length(newframe)-diff);
         tmpnf=[tmpnf;(newframe(length(newframe)-diff+1:length(newframe))...
  +temp(1:diff))./2;temp(diff+1:length(temp))];
   newframe=tmpnf;
      else
         newframe=[newframe;temp];
      end;
%
% Add new pitchperiod onto synthetic vector.
%
      y=[y;newframe];
   else
%
% Unvoiced portion of the code.  Covariance method LPC analysis is performed
% to determine vocal tract filter coefficients.  These then define a fileter
% which is excited by unit energy white noise.  Nno phase transition point
% is calculated and a single phase is assumed.
%
      numuv=numuv+1;
      if i==1,
         past=zeros(uvord,1);
      else
         past=y(length(y)-uvord+1:length(y));
      end;
```

```
      newframe=synthlpc(samples,uvord,past,0);
      numphase=[numphase 1];
%
% Add new frame to synthetic vector.
%
      y=[y;newframe];
  end;

end; % for

save twophaselpc
```

B.5 Converting Pitchmark File Into Frame Boundaries

```
% function newpfile=newpmrks(datafile, pitchfile, fs,maxperiod,uvpitch);
%
% Function: newpmrks.m
% Description: This function converts the single vector (pitchfile) containing
%              nonzero values where a closing instant was determined, into a
%              range identifying the starting point and ending point of each
%              new analysis frame/pitch period.  If there is a long gap between
%              pitch marks, the gap is broken into smaller analysis frames.
%              The size of these frames is determined by the parameters uvpitch
%              and fs.  The function will also move the starting point of each
%              frame to a point where the speech data has made a transition
%              through 0.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified:
%
% Input parameters:
% datafile: The original speech data.
% pitchfile: The vector containing nonzero values where a closing instant
%                  has been identified.
%       fs: The sampling frequency.
%       maxperiod: The size of the period (in seconds) between pitch marks above %
%       uvpitch: The frame rate to be used to subdivide long periods of speech
%                with no pitch marks.
%
% Output Parameter:
%     newpfile: A (number of frames) x 2 matrix containing the starting point
%               and ending point of each analysis frame.
%
% Subroutines directly called:
```

```
%     none
%
% Subroutines indirectly called:
%     none
%
function newpfile=newpmrks(datafile, pitchfile, fs,maxperiod,uvpitch);

%
% Identify location of each pitchmark
%
pitchmarks=find(pitchfile);
%
% Initialize some variables
%
done=0;
lastmark=1;
nextmark=0;
count=1;
total=1;
newpfile=[];
uvinc=floor(fs/uvpitch);

%
% Keep looping through until we hit every pitchmark
%
while count <= length(pitchmarks),

   nextmark=pitchmarks(count);
%
% Check to see if we have already covered the next pitch mark.  If not,
% proceed.  If so move on to next pitch mark
%
   if nextmark > lastmark

%
% Check to see if there is a large gap between pitchmarks, typically
% identifying unvoiced regions.  If not, move pitchmark to a zero-crossing.
% If so, divide into smaller analysis frames.
%
      if (nextmark-lastmark <= fs*maxperiod)

         if datafile(nextmark) > 0
            zerocrossfound=0;
            while ~zerocrossfound
       nextmark=nextmark-1;
```

```
                if datafile(nextmark) <= 0
                   zerocrossfound=1;
                end; % if datafile
             end; % while
 elseif datafile(nextmark) < 0
             zerocrossfound=0;
             while ~zerocrossfound
       nextmark=nextmark+1;
                if datafile(nextmark) >= 0
                   zerocrossfound=1;
                end; % if datafile
             end; % while
 end; % if
 newpfile(total,:)=[lastmark nextmark];
         total=total+1;
         lastmark=nextmark+1;
 count=count+1;
      else
%
% Else divide into smaller analysis frames.
%
 numsubframes=floor((nextmark-lastmark)/uvinc);
         onextmark=nextmark;
         nextmark=lastmark+uvinc;
%
% Loop through all but last subframe.
%
         for k=1:numsubframes-1,

    newpfile(total,:)=[lastmark nextmark];
            total=total+1;
            lastmark=nextmark+1;
    nextmark=nextmark+uvinc;

 end; %for k
         newpfile(total,:)=[lastmark onextmark];
         total=total+1;
         lastmark=onextmark+1;
         count=count+1;
       end; % if (nextmark-lastmark <= fs*maxperiod)

   else
      count=count+1;
   end; % if nextmark > lastmark
end; %
```

```
%
% Handle last interval (end of speech segment)
%
nextmark=length(datafile);
onextmark=nextmark;
%
% if small enough, keep as single analysis frame, otherwise divide as above
%
if (nextmark-lastmark <= fs*maxperiod)
   newpfile(total,:)=[lastmark nextmark];
else
   numsubframes=floor((nextmark-lastmark)/uvinc);
   nextmark=lastmark+uvinc;
   for k=1:numsubframes-1,
      newpfile(total,:)=[lastmark nextmark];
      total=total+1;
      lastmark=nextmark+1;
      nextmark=nextmark+uvinc;
   end; %for k
   newpfile(total,:)=[lastmark,onextmark];
   lastmark=onextmark+1;
   total=total+1;
   count=count+1;

end; % if (nextmark-lastmark <= fs*maxperiod)
```

B.6 Reading a TIMIT phoneme label file

```
% function [a,b]=readphn(phnfile);
%
% Function: readphn.m
% Description: This function reads in the hand labelled TIMIT phoneme file and
%              returns a matrix of phonmeme labels and a matrix of phoneme
%              start points and end points.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified:
%
% Input parameters:
% phnfile: The name of the TIMIT phoneme file.
%
% Output Parameters:
%     a: A matrix containing the starting point and ending points of each
```

```
%        phoneme in columns 1 and 2 respectively.  Each row is a different
%        phoneme.
%
%     b: A matrix of phoneme labels.  Each row is a different phoneme.
%
% Subroutines directly called:
%     none
%
% Subroutines indirectly called:
%     none
%
function [a,b]=readphn(phnfile);

a=[];
b=[];
%
% Open TIMIT phoneme file for reading.
%
fid=fopen(phnfile,'r');
%
% Continue to read until reaching the end-of-file.
%
while ~feof(fid)
%
% Get one line of the file as a string.
%
s=fgetl(fid);
%
% Check to see if it's the end of file, if not, continue to process.
%
if s~=(-1)
%
% Break up string into 2 integers and a string.
%
p=sscanf(s,'%i %i %s');
%
% The two integers are the start point and end point of the phoneme.
%
a=[a;p(1) p(2)];
%
% Set the numerical version of the label string to an actual string.
%
p=setstr(p(3:length(p)))';
%
% Prepend the string with spaces to bring length of string to 4 with the
```

```
% label right justified.
%
if length(p)==2
    p=['  ' p];
elseif length(p)==3
    p=[' ' p];
elseif length(p)==1
    p=['   ' p];
end;
%
% add new label to b matrix
%
b=[b;p];
end;
end;
fclose(fid);
```

B.7 Voicing determination

```
% function vuv=detvoice(curphon,phndb);
%
% Function: detvoice.m
% Description: This function determines whether the phoneme interval containing
%              the current frame of speech is voiced or unvoiced.
%
% Author: Capt Al Arb, USAF
% Date: 30 Jul 96
% Modified:
%
% Input parameters:
% curphon: The phoneme label for the current frame of speech.
%       phndb: The TIMIT phoneme data base matrix.
%
% Output Parameter:
%     vuv: voiced/unvoiced.  1=voiced, 0=unvoiced.
%
% Subroutines directly called:
%     none
%
% Subroutines indirectly called:
%     none
%
function vuv=detvoice(curphon,phndb);
```

```
vuv=0;
count=0;
done=0;
%
% Loop until we find the label
%
while ~done
   count=count+1;
%
% If the DB entry = curphon, we found it.
%
   if phndb(count,1:4)==curphon
      done=1;
      phn=count;
%
% Or if we are at the end of the file, stop and assume unvoiced.
%
   elseif count==length(phndb(:,1))
      done=1;
      phn=0;
   end;
end;
if phn
%
% If the category is VOICED, set vuv to 1.
%
if phndb(phn,5:10)=='VOICED'
   vuv=1;
end;
end;
```

B.8 Determining Transition Point

```
% function transpoint=calctrans(samples,iniord,secord,radius)
%
% Function: calctrans.m
% Description: This function estimates the transition point within
%        a voiced pitch period where the vocal tract model
%              parameters should be re-estimated.  The transition point
%              is determined by estimating the model parameters over a
%              short frame at the beginning of the pitch period, a
%              short frame at the end, and re-synthesizing the frame
%        using each set of model parameters.  Using the error
%              between the two synthetic pitch periods and the
%              original, the transition point is identified as the
```

```
%              point of intersection of the two error waveforms.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 31 Jul 96 - Added comments.
%
% Input parameters:
% samples: The original pitch period of data.
%       iniord: The analysis order to be used for the initial frame
%               parameter estimates.
%       secord: The analysis order to be used for the second frame
% parameter estimates.
%       radius: The minimum radius on the z-plane for accepting poles
% predicted using the "backward" linear prediction
% method.
%
% Output Parameters:
%       tranpoint: The point at which the vocal tract parameters should
%                  be allowed to change.
%
% Subroutines directly called:
%       calcerror.m
%
% Subroutines indirectly called:
%       none
%
function transpoint=calctrans(samples,iniord,secord,radius)

rho=[];
%
% Set size of window over which the model is estimated to 1/3 the frame length.
%
    wlengthi=floor(length(samples)/3);
    wlengths=floor(length(samples)/3);
%
% If this length is shorter than the model order desired, we need to
% increase the number of samples used for the short window.
%
    if wlengthi <= iniord
       wlengthi=min([0.8*length(samples),1.5*iniord]);
    end;

    if wlengths <= secord
       wlengths=min([0.8*length(samples),1.5*secord]);
    end;
```

```
%
% Initialize "temp" as the first window.  Prepare to estimate model over
% portion of initial phase and re-synthesize.
%
    temp=samples(1:wlengthi);

%
% Compute Polynomial Coefficients
%
    theta=getCoef(temp,iniord);
    tooshort=0;
%
% Only try to pair down poles if "getCoef" generated at least one.
%
    if (length(theta) > 1)
%
% Use "roots" to identify the actual poles corresponding to the
% coefficients.
%
       poles=roots(theta);
       numpoles=0;
%
% Look at each pole.  If its radius in the Z-plane is >= radius,
% we'll keep it. If not, reject it.  Those poles outside the unit
% circle are reflected in and those inside reflected out (1/conj()).
%
       for j=1:length(poles)
          if (abs(poles(j)) >= radius)
             numpoles=numpoles+1;
             rho(numpoles)=1/conj(poles(j));
          end; % if
       end; % for j
%
% If no poles were saved, we have a problem.  Decide to keep all outside a
% a radius of 0.5 without reflecting them (i.e. keep them stable).
%
       if numpoles==0
          sv=find(abs(poles)>=0.5);
  rho=poles(sv);
       end;

       rho=rho(:);
%
% Compute Complex Amplitudes
%
```

```
      A=getAmps(temp,rho);
%
% Generate synthesized frame.
%
      newframe_closed=reconst(A,rho,length(samples));
   else
      tooshort=1;
   end;
%
% Repeat entire process for the second phase estimating the model over a small
% section at the end of the pitch period.
%

   temp=samples(length(samples)-wlengths:length(samples));

   theta=getCoef(temp,secord);

   if (length(theta) > 1)
      poles=roots(theta);
      numpoles=0;
      for j=1:length(poles)
         if (abs(poles(j)) >= radius)
            numpoles=numpoles+1;
            rho(numpoles)=1/conj(poles(j));
         end; % if
      end; % for j
      if numpoles==0
          sv=find(abs(poles)>=0.5);
  rho=poles(sv);
       end;
       if length(rho)~=0
          rho=sort(rho(:));
          A=getAmps(temp,rho);
          newframe_open=synth4tran(A,rho,length(samples),length(temp));
       else
          tooshort=1;
       end;
   else
      tooshort=1;
   end;
%
% Calculate error between original and each synthesized frame and identify
% a transition point if one exists between 20 and 50% of the frame.
%
   if ~tooshort
```

```
      transpoint=calcerror(samples,newframe_closed,newframe_open);
   else
      transpoint=0;
   end;
```

B.9 Synthesizing Damped-Exponential Frame

```
% function [synth,poles,rho]=synthv(samples,order,radius);
%
% Function: synthv.m
% Description: This function uses three subroutines as it analyzes a frame
%              of speech and re-synthesizes it using a sum of exponentially
%              damped sinusoids as the speech model.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 30 Jul 96 - Added comments.
%
% Input parameters:
% samples: The original frame of speech.
%       order: The order for the pole determination analysis to be performed
%              over the voiced portion of "data".  This number corresponds
%              to the number of complex poles estimated.  If this is an even
%              number, there will be singord/2 complex conjugate pairs of
%              poles.
%       radius: The minimum radius on the z-plane for accepting poles
%               predicted using the "backward" linear prediction method.
%
% Output Parameters:
%     synth: The synthetic frame.
%     poles: The original complex poles estimated using the linear
%            prediction equations in the backward direction.
%     rho: The complex poles retained.  I.E. those with a magnitude (abs) >=
%          "radius".  These poles have also been reflected inside/outside
%          the unit circle.
%
% Subroutines directly called:
%     getCoef.m
%     getAmps.m
%     reconst.m
%
% Subroutines indirectly called:
%     none
%
function [synth,poles,rho]=synthv(samples,order,radius);
```

```
% initialization

y=[];
rho=[];
%
% Compute Polynomial Coefficients
%
theta=getCoef(samples,order);
%
% Only try to pair down poles if "getCoef" generated at least one.
%
if (length(theta) > 1)
%
% Use "roots" to identify the actual poles corresponding to the coefficients.
%
      poles=roots(theta);

      numpoles=0;
%
% Look at each pole.  If its radius in the Z-plane is >= 'radius', we'll keep
% it. If not, reject it.  Those poles outside the unit circle are reflected in
% and those inside reflected out (1/conj()).
%
      for j=1:length(poles)
          if (abs(poles(j)) >= radius)
            numpoles=numpoles+1;
    rho(numpoles)=1/conj(poles(j));
  end; % if (...
      end; % for j
%
% If no poles were saved, we have a problem.  Decide to keep all outside a
% a radius of 0.7 without reflecting them (i.e. keep them stable).
%
      if numpoles==0;
           sv=find(abs(poles)>=0.7);
           rho=poles(sv);
      end;
      if length(rho) ~=0
%
% Compute Complex Amplitudes
%
        A=getAmps(samples,rho);
%
% Generate synthesized frame.
```

```
%
        synth=reconst(A,rho,length(samples));
      end;

    end;
rho=rho(:);
poles=poles(:);
```

B.10 LPC synthesis

```
% function synth=synthlpc(samples,order,past,vuv);
%
% Function Name: synthlpc
% Description: This routine performs LPC analysis-synthesis on a
% frame of speech (samples) and returns a synthetic frame.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 30 Jul 96 -- Changed to all Covariance method.
%           31 Jul 96 -- Corrected energy normalization for unvoiced filter
%                        driving sequence.
%                     -- Removed filter initial conditions (Zi) from voiced
%                        filter.  Improved resulting synthesis.
%            1 Aug 96 -- Implented a straight difference equation for synthesis
%                        vice Matlab's filter command.  Now use initial
%                        conditions for all LPC synthesis.
%                     -- Added examiniation of poles for instability.
%            2 Aug 96 -- Moved pole examination to covlpc.m
%
% Input Parameters:
%     samples: The original frame of speech.
%     order: The analysis model order.
%     past: The initial conditions. i.e. the last "order" number
%           of samples from the previous frame.
%     vuv: Voiced/unvoiced flag: 0=unvoiced, nonzero=voiced.
%
% Output parameters:
%     synth: The synthetic frame.
%
% Subroutines directly called:
%     covlpc.m
%     diffeq.m
%
```

```
% Subroutines indirectly called:
%     none
%
function synth=synthlpc(samples,order,past,vuv);

%
% Check voiced/unvoiced flag
%
if vuv,

%
% get LPC coeficients and error using Covariance method LPC.
%
   [a,E]=covlpc([past;samples],order);
%
% calculate gain term as the square root of the error.
%
   G=sqrt(abs(E));
%
% Synthesize the new voiced frame using an impulse driven filter.
%
   synth=diffeq(G,a,[1;zeros(length(samples)-1,1)],past);
else % unvoiced

   [a,E]=covlpc([past;samples],order);
%
% calculate gain term as the square root of the error.
%
   G=sqrt(abs(E));

%
% Generate synthetic unvoiced frame using white noise driven filter.
%
   rndseq=randn(length(samples),1);
   rndseq=rndseq./sqrt(sum(rndseq.^2));
   synth=diffeq(G,a,rndseq,past);

end;
```

B.11 Determining Damped-Exponential Coefficients

```
% function theta=getCoef(data,order);
%
% Function: getCoef.m
% Description: This function estimates the denominator polynomial coefficients
```

```
%               for a causal system described by a set of linear equations
%               formed in the "backward" direction.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 30 Jul 96 - Added comments.
%
% Input parameters:
% data: The original frame of speech.
%       order: The order for the pole determination analysis to be performed
%              over the voiced portion of "data".  This number corresponds to
%              the number of complex poles estimated.  If this is an even
%              number, there will be singord/2 complex conjugate pairs of
%              poles.
%
% Output Parameters:
%       theta: The estimated polynomial coefficients.
%
% Subroutines directly called:
%       none
%
% Subroutines indirectly called:
%       none
%
function theta=getCoef(data,order);

%
% Initialize variables/matrices
%
data=data(:);
numsamples=length(data);

%
% Build Y matrix and b vector for backward prediction [b|A]theta=[0]
% A is the Singular Value Decomposition of Y with the diagonal elements
% of the S matrix inverted.
%
for i=1:numsamples-order,
    Y(i,:)=data(i+1:i+order)';
end;

b=-data(1:numsamples-order);

%
% Do SVD on Y
```

```
%
[U,S,V]=svd(Y);

%
% Matrix probably won't be square.  Need to invert diagonaly elements
% greater than a threshold (10^-9)
%
[numrows,numcols]=size(S);

R=min([numrows,numcols]);
for i=1:R,
    if S(i,i)>=10^(-9)
 S(i,i)=1/S(i,i);
    else
         S(i,i)=0;
    end;
end;

%
% Solve for theta.  inv(A)=VSU'.  (To be shown in appendix of thesis).
%
theta=V*S.'*(U')*b;
%
% Don't forget to tack on a 1 at the beginning!
%
theta=[1;theta];
```

B.12 Determining Damped-Exponential Complex Amplitudes

```
% function a=getAmps(data,rho);
%
% Function: getAmps.m
% Description: This function estimates the complex amplitudes associated with
%              each complex pole previously estimated from the data.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 30 Jul 96 - Added comments.
%
% Input parameters:
% data: The original frame of speech.
%       rho: The complex poles previously estimated from the data.
%
% Output Parameters:
%       a: The estimated complex amplitudes.
```

```
%
% Subroutines directly called:
%       none
%
% Subroutines indirectly called:
%       none
%
function a=getAmps(data,rho);

%
% Initialize variables and matrices
%
data=data(:);
numsamples=length(data);
rho=rho(:);
R=ones(numsamples,length(rho));

%
% Build R matrix for equation Ra=d
%
for i=1:numsamples-1,
   R(i+1,:)=(rho').^i;
end;

%
% Solve for a
%

a=pinv(R)*data;
```

B.13 Damped-Exponential Re-synthesis Routine

```
% function Y=reconst(amp,rho,numsamples);
%
% Function: reconst.m
% Description: This function synthesizes the new frame of speech using the complex
%              poles and amplitudes previously estimated as the parameters for the
%              exponentially damped sinusoids.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 30 Jul 96 - Added comments.
%
% Input parameters:
% amp: The complex amplitudes previously estimated from the data.
```

```
%       rho: The complex poles previously estimated from the data.
%       numsamples: The number of samples to synthesize.
%
% Output Parameters:
%       Y: The synthetic frame
%
% Subroutines directly called:
%       none
%
% Subroutines indirectly called:
%       none
%
function Y=reconst(amp,rho,numsamples);

%
% initialization
%
amp=amp(:);
rho=rho(:);
R=ones(numsamples,length(rho));

%
% Build R matrix for summation.  Y=Ramp where
%      |rho(0)^0   rho(1)^0   ... rho(k-1)^0  |
%      |rho(0)^1   rho(1)^1   ... rho(k-1)^1  |
%  R=  |   :           :       :      :       |
%      |rho(0)^n-1 rho(1)^n-1 ... rho(k-1)^n-1|
%

for i=0:numsamples-1,
   R(i+1,:)=(rho').^i;
end;

% Solve equation

Y=R*amp;
Y=real(Y);
```

B.14 Determining Error in Transition Point Estimation

```
% function point = calcerror (original,closed,open);
%
% Function: calcerror.m
% Description: Using the error between the two synthetic pitch periods
```

```
%              and the original, the transition point is identified as the
%              point of intersection of the two error waveforms.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 31 Jul 96 - Added comments.
%
% Input parameters:
% original: The original pitch period of data.
%       closed: The synthetic pitch period generated using parameters
% estimated over the small window at the beginning of
%         the pitch period.
%         open: The synthetic pitch period generated using parameters
% estimated over the small window at the beginning of
%         the pitch period.
%
% Output Parameters:
%        point: The point at which the vocal tract parameters should
%               be allowed to change.
%
% Subroutines directly called:
%  none
%
% Subroutines indirectly called:
%        none
%
function point = calcerror (original,closed,open);


%-------------------------------------
% Peak finding/curve fitting approach
%-------------------------------------
% Connect the peaks of the error signal to get smooth error waveform.
%

clerror=((original-closed).^2)./sum((original-closed).^2);
%
% w is the number of points on each side of the center of the analysis
% window used for determining peaks.
%
w=2;
thresh=10^(-8)*max(abs(clerror));
newclerror=clerror;
%
% Loop throut
```

```
for j=w+1:length(clerror)-w;

%
% Find the peak in the analysis window.
%
   [peak, peakbin]=max(abs(clerror(j-w:j+w)));
   peak=peak(1);
   peakbin=peakbin(1);
   if peak > thresh,
%
% Keep the max point and set all others to zero.
%
      if peakbin==1,
         newclerror(j-w+1:j+w)=zeros(2*w,1);
      elseif peakbin == 2*w+1,
         newclerror(j-w:j+w-1)=zeros(2*w,1);
      else
         newclerror(j-w:j-w+peakbin-2)=zeros(peakbin-1,1);
 newclerror(j-w+peakbin:j+w)=zeros(2*w+1-peakbin,1);
      end;
   end;
end;

%
% Now all we have in the error vector is the peaks.  Linearly
% interpolate lines between each one.
%
peaks=find(newclerror);
newclerror(1:peaks(1))=linspace(0,newclerror(peaks(1)),peaks(1));
if length(peaks)>2
  for k=2:length(peaks),
      newclerror(peaks(k-1):peaks(k))=linspace(newclerror(peaks(k-1)),...
   newclerror(peaks(k)),peaks(k)-peaks(k-1)+1);
  end;
else k=2;
end;
newclerror(peaks(k):length(newclerror))=linspace(newclerror(peaks(k)),...
          newclerror(length(clerror)),length(clerror)-peaks(k)+1);

%
% Repeat process for the other synthetic frame.
%

operror=((original-open).^2)./sum((original-open).^2);
w=2;
```

```
thresh=10^(-8)*max(abs(operror));
newoperror=operror;
for j=w+1:length(operror)-w;

   [peak, peakbin]=max(abs(operror(j-w:j+w)));
   peak=peak(1);
   peakbin=peakbin(1);
   if peak > thresh,
      if peakbin==1,
            newoperror(j-w+1:j+w)=zeros(2*w,1);
      elseif peakbin == 2*w+1,
            newoperror(j-w:j+w-1)=zeros(2*w,1);
      else
            newoperror(j-w:j-w+peakbin-2)=...
zeros(peakbin-1,1);
    newoperror(j-w+peakbin:j+w)=...
zeros(2*w+1-peakbin,1);
end;
   end;
end;

peaks=find(newoperror);
newoperror(1:peaks(1))=linspace(0,newoperror(peaks(1)),peaks(1));
if length(peaks)>2
  for k=2:length(peaks),
    newoperror(peaks(k-1):peaks(k))=linspace(newoperror(peaks(k-1)),...
newoperror(peaks(k)),peaks(k)-peaks(k-1)+1);
   end;
else k=2;
end;
if length(peaks)>=2,
   newoperror(peaks(k):length(newoperror))=linspace(newoperror(peaks(k)),...
          newoperror(length(newoperror)),length(operror)-peaks(k)+1);
elseif length(peaks)==1
   newoperror(1:peaks(1))=linspace(newoperror(1),newoperror(peaks(1)),peaks(1));
   newoperror(peaks(1)+1:length(newoperror))=linspace(newoperror(peaks(k)+1),...
       newoperror(length(newoperror)),length(newoperror)-peaks(1)+1);
end;
%
% Define the window over the entire pitch period that we will look for
% a crossing point.  Define the window as 20%-50% of the period.
%
swin=ceil(0.2*length(clerror));
ewin=ceil(0.6*length(clerror));
k=swin;
```

```
if k < 20
   k=20;
end;

crossfound=0;
point=0;

while (~crossfound)&(k<=ewin)

%
% If the error from the period generated from the end-of-frame
% parameters, is less than the that generated from the start-of-frame
% parameters, we've found the point.  Otherwise, keep searching.
%
    if newoperror(k) <= newclerror(k)
        point=k;
crossfound=1;
    else
k=k+1;
    end;
end;

%
% If no point was found, define the transition point as the end of
% the search window.
%
if ~point
   point=ewin;
end;
```

B.15 Re-synthesizing in Transition Point Estimation

```
% function Y=synth4tran(amp,rho,numsamples,mlength);
%
% Function: synth4tran.m
% Description: This function synthesizes the new frame of speech using the
%              complex poles and amplitudes previously estimated as the
%              parameters for the exponentially damped sinusoids.  However, it
%              assumes that the time=0 point is point numsamples-mlength+1 and
%              that the time indices are negative below this point.
%
% Author: Capt Al Arb, USAF
% Date: 29 Jul 96
% Modified: 31 Jul 96 - Added comments.
```

```
%
% Input parameters:
% amp: The complex amplitudes previously estimated from the data.
%       rho: The complex poles previously estimated from the data.
%       numsamples: The number of samples to synthesize.
%       mlength: The length of the frame the model parameters were
%                estimated over.
%
% Output Parameters:
%       Y: The synthetic frame
%
% Subroutines directly called:
%       none
%
% Subroutines indirectly called:
%       none
%
function Y=synth4tran(amp,rho,numsamples,mlength);


% Initialize vectors/matrices

amp=amp(:);
rho=rho(:);
R=ones(numsamples,length(rho));
spoint=mlength-numsamples+1;

% Build R matrix alowing for initial point to be other than 1
% and endpoint to be other than the end of the model frame.

for i=spoint:mlength,
   R(i+abs(spoint)+1,:)=(rho').^i;
end;

%
% Solve for Y
%
Y=R*amp;
Y=real(Y);
```

B.16 Difference Equation Implementation for LPC synthesis

```
% function y=diffeq(B,A,X,ic);
%
```

```
% Function: diffeq.m
% Description: This function implements the difference equation:
%
% y(n)=B(1)*X(1)-A(n-1)*y(n-1)-A(n-2)*y(n-2)-...-A(n-p)*y(n-p)
%
% where p is the number of coefficients in the denominator polynomial.
%
% Author: Capt Al Arb, USAF
% Date: 1 Aug 96
% Modified:
%
% Input parameters:
% B: The numerator coefficient, typically the Gain for LPC synthesis.
%       A: The denominator polynomial coefficients.
%       X: The filter driving function.
%      ic: The initial conditions of the filter.
%
% Output Parameter:
%       y: The filter output.
%
% Subroutines directly called:
%     none
%
% Subroutines indirectly called:
%     none
%

function y=diffeq(B,A,X,ic);

y=ic;
B=B(:);
A=A(:);
X=X(:);
ic=ic(:);
for n=length(ic)+1:length(ic)+length(X),
   y(n)=(B(1)*X(n-length(ic))-A(2:length(A))'*y(n-1:-1:n-length(A)+1))/A(1);
end;
y=y(length(ic)+1:length(y));
```

B.17 Routine for Two-Phase LPC Synthesis

```
% function synth=synthlpc2(samples,order,past,vuv,phase);
%
% Function Name: synthlpc
% Description: This routine performs LPC analysis-synthesis on a
```

```
% frame of speech (samples) and returns a synthetic frame.
%
% Author: Capt Al Arb, USAF
% Date:  2 Aug 96
% Modified:
%
% Input Parameters:
%     samples: The original frame of speech.
%     order: The analysis model order.
%     past: The initial conditions. i.e. the last "order" number
%           of samples from the previous frame.
%     vuv: Voiced/unvoiced flag: 0=unvoiced, nonzero=voiced.
%     phase: 1=initial phase of voiced speech, 2=second phase.
%
% Output parameters:
%     synth: The synthetic frame.
%
% Subroutines directly called:
%     covlpc.m
%     diffeq.m
%
% Subroutines indirectly called:
%     none
%
function synth=synthlpc2(samples,order,past,vuv,phase);

%
% Check voiced/unvoiced flag
%
if vuv,

%
% get LPC coeficients and error using Covariance method LPC.
%
   [a,E]=covlpc([past;samples],order);
%
% calculate gain term as the square root of the error.
%
   G=sqrt(abs(E));
%
% Synthesize the new voiced frame using an impulse driven filter or no
% input depending on phase.
%
   if phase==1,
      synth=diffeq(G,a,[1;zeros(length(samples)-1,1)],past);
```

```
   else
      synth=diffeq(G,a,zeros(length(samples),1),past);
   end;


else % unvoiced

   [a,E]=covlpc([past;samples],order);
%
% calculate gain term as the square root of the error.
%
   G=sqrt(abs(E));

%
% Generate synthetic unvoiced frame using white noise driven filter.
%
   rndseq=randn(length(samples),1);
   rndseq=rndseq./sqrt(sum(rndseq.^2));
   synth=diffeq(G,a,rndseq,past);

end;
```

Bibliography

1. "The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus, NIST Speech Disc CD1-1.1." Produced on CD-ROM by the National Institute of Standards and Technology (NIST), October 1990.
2. Abe, Masanobu. "A Segment-Based Approach to Voice Conversion," *Proc. 1991 Int. Conf. Acoust. Speech Signal Process.*, *2*:765–768 (May 1991).
3. Cummings, K. and M. Clements. "Glottal Models for Digital Speech Processing: A Historical Survey and New Results," *Digital Signal Processing*, *5*:21–42 (1995).
4. Entropic Research Laboratories, Inc. *ESPS Programs A-L* (Version 5 Edition), 1993.
5. Fant, G. and others. "A Four Parameter Model of Glottal Flow," *Speech Transmission Laboratory Quarterly and Status Report*, 1–14 (Oct.-Dec. 1985).
6. Flanagan, J.L. "Phase Vocoder," *J. Acoust. Soc. Am.*, *51*:1375–1387 (1972).
7. Fujisaki, H. and M. Ljungqvist. "Proposal and Evaluation of Models for the Glottal Source Waveform." *Proc. 1986 Int. Conf. Acoust. Speech Signal Process.*. 1605–1608. 1986.
8. Hedelin, P. "A Glottal LPC-Vocoder." *Proc. 1984 Int. Conf. Acoust. Speech Signal Process.*. 1.6.1–1.6.4. 1984.
9. Hedelin, P. "High Quality Glottal LPC-Vocoding." *Proc. 1986 Int. Conf. Acoust. Speech Signal Process.*. 465–468. 1986.
10. Holmes, J. "The Influence of Glottal Waveform on the Naturalness of Speech from a Parallel Formant Synthesizer," *IEEE Trans. Aud. Electroacoust.*, *21*(3):298–305 (1973).
11. Holmes, J. and others. "Speech Synthesis By Rule," *Language and Speech*, *7*:127–143 (1964).
12. John R. Deller, et al. *Discrete-Time Processing of Speech Signals*. New York: Macmillan Publishing Company, 1993.
13. Keppel, Geoffrey. *Design and Analysis: A Researcher's Handbook* (Third Edition). Englewood Cliffs, New Jersey: Prentice Hall, 1991.
14. Klatt, D.H. and L.C. Klatt. "Analysis, Synthesis, and Perception of Voice Quality Variations Among Male and Female Talkers," *J. Acoust. Soc. Am.*, *87*:820–857 (1990).
15. Krishnamurthy, Ashok K. "Glottal Source Estimation Using a Sum-of-Exponentials Model," *IEEE Trans. Signal Process.*, *40*(3):682–686 (March 1992).

16. Kroon, Peter and Bishnu S. Atal. "Predictive Coding of Speech Using Analysis-by-Synthesis." *Advances in Speech Signal Processing* edited by Sadaoki Furui and M. Mohan Sondhi, New York: Marcell Dekker, Inc., 1992.

17. Kumaresan, R. "On the Zeros of the Linear Prediction-Error Filter for Deterministic Signals," *IEEE Trans. Acoust. Speech Signal Process.*, *31*(1):217–220 (February 1983).

18. Kumaresan, R. and D. Tufts. "Estimating the Parameters of Exponentially Damped Sinusoids and Pole-zero Modeling in Noise," *IEEE Trans. Acoust. Speech Signal Process.*, *30*(5):833–840 (December 1982).

19. Mammone, Richard J. and others. "Robust Speaker Recognition: A Feature Based Approach," *IEEE Signal Processing Magazine*, *13*(5):58–71 (September 1996).

20. McAulay, Robert J. and Thomas F. Quatieri. "Speech Analysis/Synthesis Based on a Sinusoidal Representation," *IEEE Trans. Speech and Audio Process.*, *34*(5):744–754 (August 1986).

21. McAulay, Robert J. and Thomas F. Quatieri. "Low-Rate Speech Coding Based on the Sinusoidal Model." *Advances in Speech Signal Processing* edited by Sadaoki Furui and M. Mohan Sondhi, New York: Marcell Dekker, Inc., 1992.

22. McMillan, Capt Vance M. *Rule-Based Frequency Domain Speech Coding*. MS thesis, AFIT/GE/ENG/90D-38, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, December 1990.

23. O'Shaughnessy, Douglass. "Approaches to Improve Automatic Speech Synthesis." *Modern Methods of Speech Processing* edited by Ravi P. Ramachandran and Richard J. Mammone, Boston: Kluwer Academic Publishers, 1995.

24. Parthasarathy, S. and D. Tufts. "Excitation-synchronous Modeling of Voiced Speech," *IEEE Trans. Acoust. Speech Signal Process.*, *35*:1241–1249 (September 1987).

25. Pellissier, Stephen V., CPT (USA). *Text-Independent, Open-Set Speaker Recognition*. MS thesis, AFIT/GE/ENG/96M-01, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, March 1996.

26. Rabiner, Lawrence and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice Hall, 1993.

27. Rosenberg, A. "Effect of the Glottal Pulse Shape on the Quality of Natural Vowels," *J. Acoust. Soc. Am.*, *49*(2):583–590 (1971).

28. Secrest, Bruce G. and George R. Doddington. "An Integrated Pitch Tracking Algorithm for Speech Systems," *Proc. 1983 Int. Conf. Acoust. Speech Signal Process.*, *3*:1352–1355 (1983).

29. Siegel, Sidney and N. John Castellan Jr. *Nonparametric Statistics for the Behavioral Sciences* (Second Edition). New York: McGraw Hill Book Co., 1988.

30. Sproat, Richard W. and Joseph P. Olive. "Text-to-Speech Synthesis," *AT&T Technical Journal*, *74*(2):35–44 (March/April 1995).

31. Therrien, Charles W. *Discrete Random Signals and Statistical Signal Processing.* Englewood Cliffs, New Jersey: Prentice Hall, 1992.

Vita

H. Allan Arb was born [redacted] He graduated from Savanna High School, Savanna, Illinois in 1987. In late June, 1987, he entered the United States Air Force Academy. In 1991, he received the Bachelor of Science in Electrical Engineering degree and was commissioned a second lieutenant in the Air Force. He left Colorado Springs for his first assignment at the Air Force Information Warfare Center (AFIWC) at Kelly Air Force Base, San Antonio, TX. His first position at the AFIWC was as a radar parametrics engineer responsible for the analysis and technical documentation of radar systems owned and operated by the United States. He was then appointed as the head of the Radar Section, in the Engineering Data Division, AFIWC. In 1993, then 1st lieutenant Arb, was put in charge of the Technical Development Branch, Engineering Data Base Division, C2W Information Directorate. In May, 1995, he was promoted to Captain and transferred to the Air Force Institute of Technology, Wright-Patterson AFB, Ohio to pursue the Master of Science degree in Electrical Engineering. He is a member of the Tau Beta Pi and Eta Kappa Nu honor societies. Upon graduation, Captain Arb will continue his education in pursuit of the Doctor of Philosophy degree in Electrical Engineering at AFIT.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1996	Master's Thesis

4. TITLE AND SUBTITLE
A Two-Phase Damped-Exponential Model for Speech Synthesis

5. FUNDING NUMBERS

6. AUTHOR(S)
H. Allan Arb
Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
2950 P Street
Wright-Patterson Air Force Base, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/96D-02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AL/CFBA (Dr. Raymond Slyh)
Building 441
2610 Seventh Street
Wright-Patterson AFB OH 45433-7901

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

It is well known that there is room for improvement in the resultant quality of speech synthesizers in use today. This research focuses on the improvement of speech synthesis by analyzing various models for speech signals. An improvement in synthesis quality will benefit any system incorporating speech synthesis. Many synthesizers in use today use linear predictive coding (LPC) techniques and only use one set of vocal tract parameters per analysis frame or pitch period for pitch-synchronous synthesizers. This work is motivated by the two-phase analysis-synthesis model proposed by Krishnamurthy. In lieu of electroglottograph data for vocal tract model transition point determination, this work estimates this point directly from the speech signal. The work then evaluates the potential of the two-phase damped-exponential model for synthetic speech quality improvement. LPC and damped-exponential models are used for synthesis. Statistical analysis of data collected in a subjective listening test indicates a statistically significant improvement (at the 0.05 significance level) in quality using this two-phase damped-exponential model over single-phase LPC, single-phase damped-exponential, and two-phase LPC for the speakers, sentences, and model orders used. This subjective test shows the potential for quality improvement of synthesized speech and supports the need for further research and testing.

14. SUBJECT TERMS
Speech, Speech Synthesis, Damped-Exponential, LPC, Analysis-Synthesis, ANOVA, Analysis of Variance, Human Subject Testing, Speech Modeling

15. NUMBER OF PAGES
105

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL